Final Masters

Master's degree in Automatic Control and Robotics

Mechanical and control design of an active coupling for a teleoperated surgical instrument

Author: Sergio Sánchez Gallego

Director: Alícia Casals

Codirector: Eduard Bergés

Convocation: april 2016

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE MÀSTER **Curs: 2014/15**

Informe de lliurament:

El Director /ponent dóna el seu vistiplau al lliurament del TFM referenciat a continuació per a la seva presentació i defensa

TITULACIÓ: Master's degree in Automatic Control and Robotics		
Nom estudiant: Sergio Sánchez Gallego		
DNI/Passaport: 52429790W	**Telèfon:** 639791682	
Mail: sersangal@gmail.com		
Realitzat en una empresa	**Nom de l'empresa**	**Director**
En cas de fer el TFM compartit	**Nom estudiant:**	**DNI/Passaport**

Títol del TFM: Mechanical and control design of an active coupling for a teleoperated surgical instrument	
Director/ponent: Alícia Casals Gelpí	
Departament: ESAII	
Idioma de la memòria: Anglès	**Idioma en què es farà la presentació:** Català
Convocatòria: 1ª Extraordinaria	

Lliurament:

Data de lliurament: 15-4-2016	**Signatura:** 

Proposta de Tribunal (opcional)	**Noms**
President	**Antonio B Martínez**
Vocal	**Joan Aranda**
Vocal (El mateix director)	**Alicia Casals**

Barcelona, a 15 d'abril de 2016

Autorització per a la difusió de treballs acadèmics (PFC, TFG, TFM, etc.) de l'ETSEIB a través del dipòsit institucional *UPCommons*

Difusió pública del treball acadèmic

En Sergio Sánchez Gallego DNI núm. 52429790W autor/a del treball acadèmic titulat Mechanical and control design of an active coupling for a teleoperated surgical instrument **autoritzo** la comunicació pública de les dades bibliogràfiques i del text complet del treball en xarxa a través del dipòsit institucional *UPCommons* o plataforma que el substitueixi. Aquesta difusió només es portarà a terme sempre i quan el corresponent professor director, coordinador o tutor n'hagi descartat un tractament confidencial.

Atorgament d'una llicència Creative Commons

*[ompliu aquest apartat només si **heu autoritzat** la difusió del text complet del treball en xarxa]*
El sotasignant autoritza la difusió del treball acadèmic:

> mitjançant la llicència CC "**Reconeixement-NoComercial-SenseObraDerivada**"
> [permet reproduir, distribuir, comunicar públicament però no fer obres derivades (traduccions, etc.), sempre i quan s'esmenti l'autoria i no es facin usos comercials]

Difusió pública d'una adreça de correu electrònic

El sotasignant autoritza la difusió del registre bibliogràfic del seu treball:

> **amb l'adreça de correu electrònic** sersangal@gmail.com
> [s'ofereix una adreça de contacte que permeti la futura comunicació entre l'autor i els investigadors, empresaris o altres possibles usuaris interessats en la seva obra]

En cap cas aquesta autorització implica una cessió en exclusiva dels drets d'explotació de l'autor/a sobre l'obra ni impedeix l'explotació normal de l'obra a través de les formes habituals.

La durada de la cessió de drets serà indefinida i pot ésser terminada per denúncia unilateral de l'autor/cedent, per terminació d'ambdues parts de mutu acord o per incompliment de qualsevol de les parts de les obligacions contingudes a la mateixa.

L'autor/a declara que és el legítim propietari dels drets d'autor de l'obra que s'autoritza. Si el document inclou obres de les quals no n'és el propietari dels drets d'explotació (fotografies, dibuixos, textos, etc.), l'autor/a declara que ha obtingut el permís sense restricció del titular corresponent per atorgar la present autorització.

Barcelona, 15 d'abril de 2016 Signatura de l'autor/a:

SUMMARY

This project is oriented to design and implement a mechanical coupling of surgical instruments for laparoscopy. Laparoscopy to a robot, a Minimally Invasive Surgery (MIS) technique, is a surgical technique where the surgeon is capable to carry out an internal operation through small incisions (0.5-1.5 cm) in the patient's body. The inclusion of new technologies in this field has improved the MIS technique [1].

Most of the laparoscopy instruments have limited degrees of freedom (DOF) which is a limitation when the movements done by the surgeon are inside the abdominal wall of the patients. Increasing the number of DOF for laparoscopy instruments avoid this limitation.

The aim of this project is to increase the dexterity of two teleoperated robots, which are installed in the ESAII LAB with a manual surgical instrument attached to their end effector. The instruments themselves are provided with 1 DOF at the tip, which is actuated by a servo to open or close the clamps. The robots add six additional DOF to the tool, three for the spatial positioning and three for the orientation of the tool. The main handicap is that these DOF are located at the handle of the instrument, (the last joint of the robot), and transferring them to the tool's tip may require large movements of the robots due to geometrical constraints. In some cases, the reverse kinematics needed leads the robot to a joints links configuration unavailable inside the working range. To achieve the objective, three degrees of freedom instruments for laparoscopy will be coupled to our robots' end effectors which facilitates the orientation of the tool without the need for the robot to perform large displacements. Therefore, the mechanical parts and the electronic board will be designed in order to attach this instrument to our robots and be able to control them through teleoperation.

The tip of each instrument has 3 DOF mechanically driven, so an active transmission is required in order to move them for the tool's tip orientation. The robots' end effectors has been modified to increase the handling of the actual surgical instrument. Our work consists in designing and developing the physical coupling between robot and its surgical instrument, the servos coupling and also the control of the servos which provide the orientation for the tool tip.

INDEX

1. GLOSSARY

CCW: Counterclockwise

CPU: Central Processing Unit or

CW: Clockwise

DOF: Degree Of Freedom

DXL: DYNAMIXEL

FDM: Fused Deposition Modeling

HD-ASC: Half Duplex Asynchronous Serial Communication

I/O: Input/output

IP: Internet Protocol

MCP: Manual Control Pendant.

MIRS: Minimally Invasive Robotic Surgery

MIS: Minimally Invasive Surgery

PCB: Printed Circuit Board

TCP: Transmission Control Protocol

TCP/IP: Communication protocol used on the Internet and similar computer networks

UART: Universal Asynchronous Receiver/Transmitter

USB: Universal Serial Bus

2. PREFACE

The aim of this project is to improve the capabilities of two robots installed at the ESAII's lab for research in the field of robotic surgery. The robots are setup to simulate laparoscopic surgery and the maneuverability of the tools inside the patient is critical. Our objective consists of increasing the available DOF at the tip of the tools by adapting mechanical and more sophisticated surgical instruments, including the design and implementation of the electronics for the teleoperation control.

2.1. Project Origin

The initiative of this project stems from the need to improve the dexterity of two robots for laparoscopic surgery, which are provided with surgical instruments with only one DOF. The idea is to endow these robots with surgical instruments with 3 DOF at the tip, so that more complex tasks can be performed.

2.2. Motivation

With the increase of robotic applications, robots are being introduced to many fields with the aim to help humans and help people to perform more and more complex tasks.

Different fields of engineering are integrated to increasingly develop complex systems for people's health purposes. One of the fields where robotics are becoming more and more present is in surgery (Figure 1), as it is the case of MIRS (Minimal Invasive Robotic Surgery) [2]. Although robotic surgery is sometimes questioned [3].

There exist different instruments in laparoscopy with different mechanical structures but sometimes they don't offer an ergonomic structure for the surgeon. The inclusion of robots in this fields help to surgeons to perform complex tasks thanks to including instruments with additional DOF at the tip.

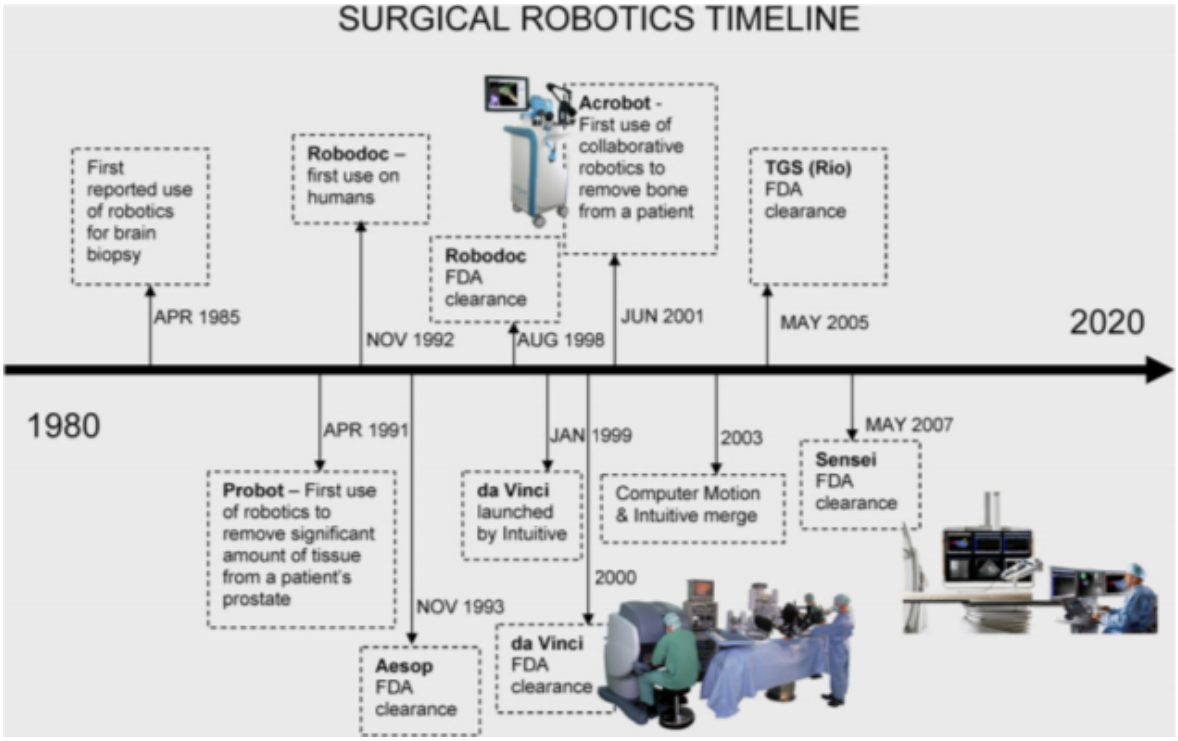


Figure 1. Surgical robotics timeline [4].

2.3. Prerequisites

In order to carry out this project, one of our 6 DOF STÄUBLI robots (Figure 2) has been used. A surgical instrument with 3 DOF at the tip has been mounted at the robot's end effector.

Figure 2. STÄUBLI RX60B robot.

Concretely, the instrument that has been attached is a grasping forceps with 3 DOF.

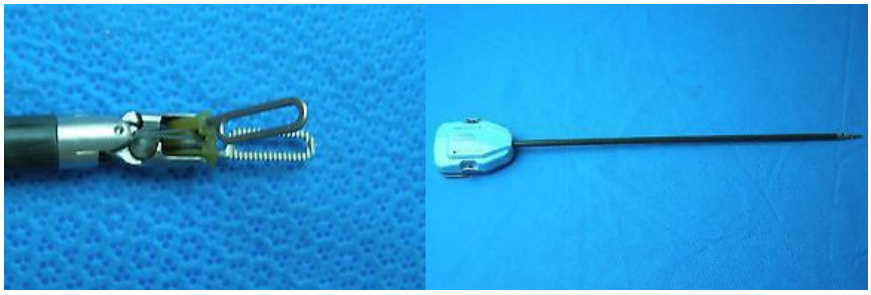


Figure 3. Da Vinci forceps

In this project, the robots are controlled through a 6 DOF master device, a *Phantom Omni haptic device* (Figure 4). This device will be used to test the control of the *da Vinci* forceps. To carry out this test, 3 DOF from the device will be transferred to the robot in order to determine the spatial positioning, while the others 3 DOF will determine the orientation of the forceps.

Figure 4. PHANTOM® Omni™ Force Feedback Joystick

3. INTRODUCTION

Laparoscopy surgery, also called Minimal Invasive Surgery (MIS), is the surgical operation done through small incisions with the help of special surgical instruments and endoscopes. It has advantage for patients, but also has disadvantage for surgeons. The patient benefits are various, such as reduction of risk, disfigurements and patient pain shorter immobilization and an earlier return to work, with reduced time and cost of postoperative care [5].

The laparoscopy disadvantages are for surgeons reduced sight and pressure, and especially when exist motion restriction because of pivot point (trocar kinematic), reduced tactile because of long instruments and amplification of the tremor because of the large lever arm [6].

Commonly, MIS instruments only have 1 DOF at the tip limiting the movement inside the abdomen. While surgeons use this kind of instruments, they are forced to make more movements with their wrist in order to produce the desired movement. It can be frustrating in some cases because they have also movement limitations, lack of ergonomics, hang fatigue and, muscular pain can take place.

The use of instruments with 3 DOF at the tip, inside the abdomen could solve these disadvantages. There is one robotic system which performs laparoscopic surgeries through teleoperation, i.e. the *da Vinci* robot, which has proved to offer benefits such as improving dexterity and precision.

We want to improve our robots dexterity, by means of instruments endowed with 3 DOF in the tip, so that complex tasks can be performed. This project consist of designing and developing the coupling servos which motorize these instruments, including the required electronics, the couplings which mount the tweezers to our robots and the software needed to teleoperate both the robot and the tweezers through the *Phantom haptic device* to carry out the first tests.

3.1. Project objectives

The main objective of this project is to replace the current surgical instruments attached to our robots (instruments with 1 DOF at the tip) by the motorized *da Vinci* forceps. We expect to overcome the formers maneuverability drawbacks by fitting the new instruments with 3 DOF at their tips.

3.2. Project scope

This project focuses on getting a robot operating with the new instrument 100% functional to perform teleoperated laparoscopy experiments. In order to achieve these objectives, an electronic board will be designed and also the mechanical coupling to assembly the instruments to the robot. As this is not a theoretical project, the mechanical parts will be manufactured by 3D printer and the electronic board will be assembled ready to be kicked in action the full system. Also, in order to carry out the teleoperated control a communication protocol will be designed to transfer the forceps' tip orientation from a *Phantom haptic device*.

4. STATE OF THE ART

This chapter explains the state of the art of laparoscopic surgeries and the surgical instruments used in laparoscopic so far. Our work is also put into context, so that the reader understands better the difficulties and needs of this field.

4.1. Laparoscopic Surgery

Laparoscopy is a surgical technique where the surgeon is capable to carry out an internal operation through small incisions in the patient's body. The patient's abdomen is insufflated with cannulas (essentially metal tubes, but also can be found in plastic) in which pneumatic check valves are passed through small incisions to provide entry ports for laparoscopic surgical instruments and cameras, needed to visualize the surgical zone (Figure 5).

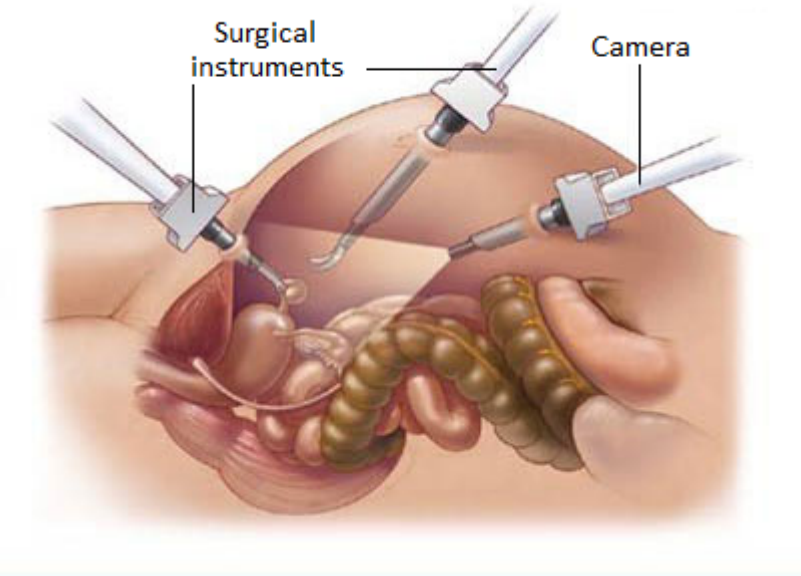


Figure 5. Illustration laparoscopic surgery.
Usually 3 incisions, 2 surgical instruments and 1 for the camera.

Advances in technology have led systems and tools to be improved, so that it is possible to perform such operations with faster recovery and reduced bleeding and less invasive procedures (Figure 6). Moreover, there is a cost reduction associated with hospital charges.

Figure 6. Visible results of a surgical operation by laparoscopy.
The laterals incisions were used for the surgical instruments.
There is a third incision in the belly button for the camera which is less visible.

4.2. Instruments in laparoscopy

Many new surgical instruments have been developed to proceed with different tasks in the field of MIS. The most common instruments are graspers and dissectors.

Laparoscopic instruments have evolved through three basic generations:

- First generation: Laparoscopic instruments cannot be disassembled and are the most difficult to clean.
- Second generation: Laparoscopic instruments cannot be disassembled, but have a port which facilitates their cleaning.
- Third generation: Laparoscopic instruments can be disassemble into two or more pieces for cleaning.

The new surgical instruments are more ergonomic, have a wider range of motion and a more accurate motion control. Hand held instruments in laparoscopic surgery provide the same basic function as open surgical instruments.

An important concept to mention is the available DOF on the instrument or the devices used during surgical interventions. The more available DOF a surgical instrument has, the better can surgeons orientate the end effector and reach places of difficult access.

Laparoscopic instruments are made of durable materials, usually high quality stainless steel. The most conventional instruments used in MIS are hand-held instruments with long shafts, the end effector of which can be a needle holder, forceps, dissectors etc. *Figure* 7 shows an example set of end effectors used in laparoscopic surgery.

The instrument is inserted into the patient through a trocar which is well constrained by a pivoting point. A trocar is shaped like a pen with a sharp ending to facilitate the pass through

patients' skin.

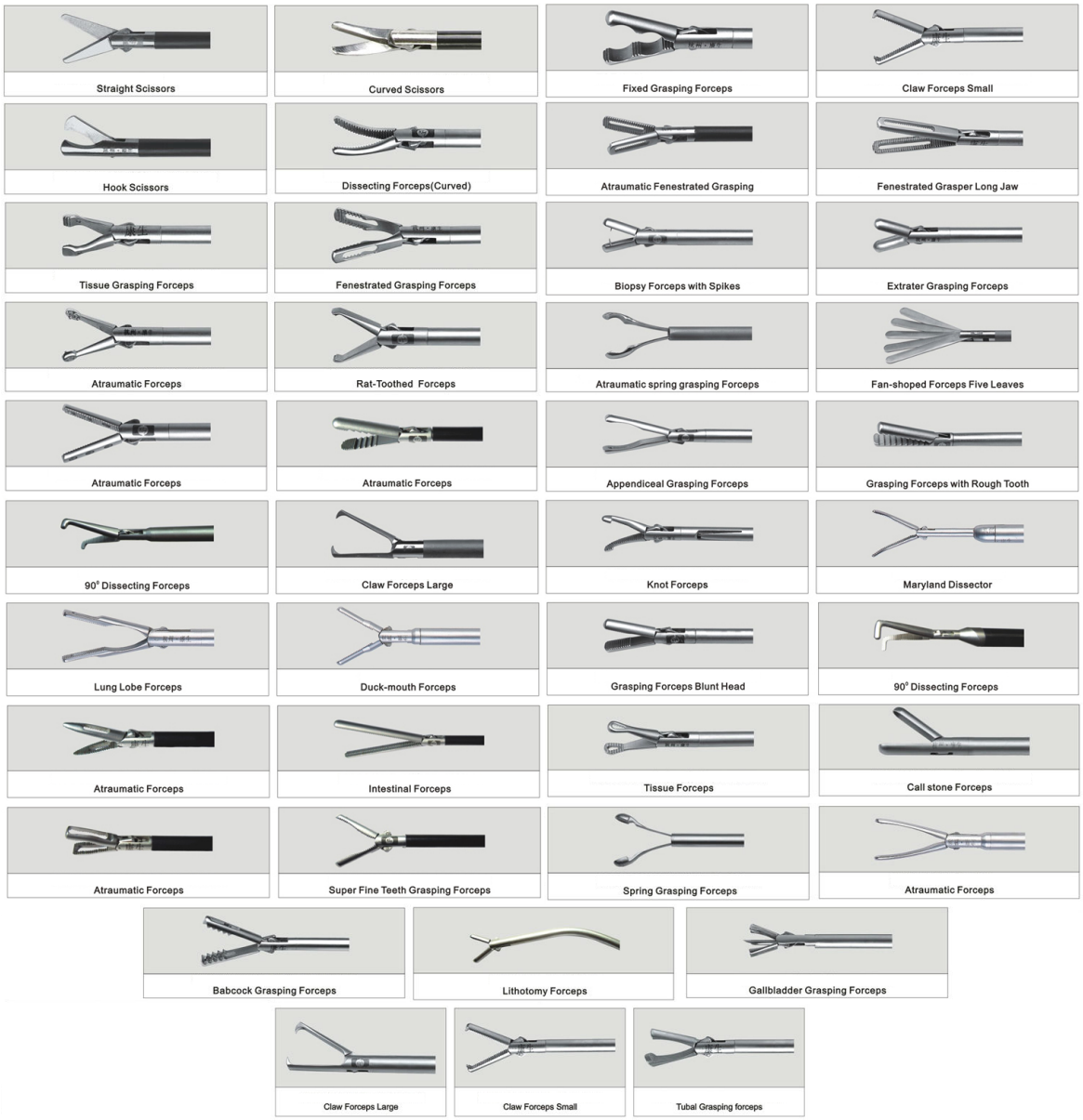


Figure 7. Examples of laparoscope surgery instruments.

4.2.1. Conventional surgical instruments

Conventional surgical instruments for laparoscopy commonly dispose of 1 DOF at the tip (Figure 8). They consist of a handle, a knob and a shaft (Figure 9). The knob located between the handle and the shaft, can be moved allowing the shaft rotate 360º to both sides (CW and CCW).

This type of instruments are simpler and easy to use but they generate discomfort along the surgeries by the lack of ergonomics. Due to their simplicity, they have a 5 mm diameter shaft (this means less weight and the incisions on the patients are smaller) and can perform any phase of the surgical intervention, but at the same time they require more hand movements to perform the surgical task.

Figure 8. Conventional laparoscopic instrument disassembled sequence.

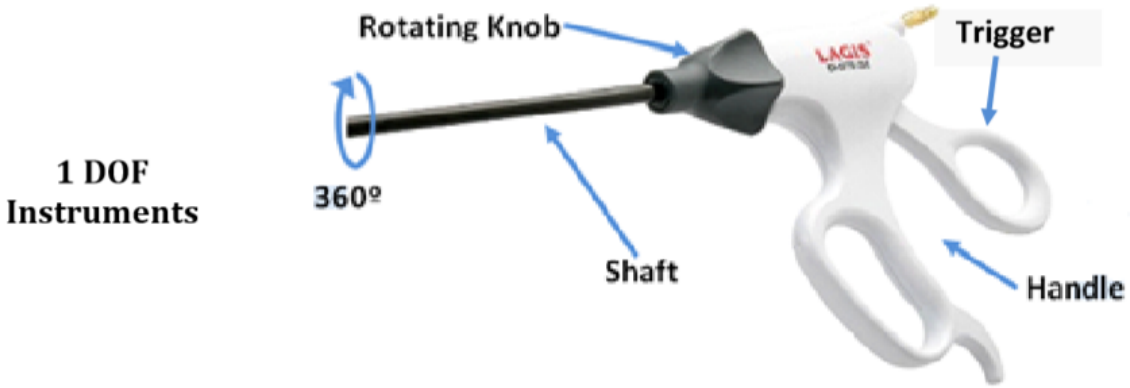


Figure 9. Conventional laparoscopic instrument with 1 DOF at the tip.

4.2.2. Articulated surgical instruments

An articulated instrument for laparoscopy (Figure 10) usually has between 2 and 3 DOF. These instruments have more complex transmission systems to orientate the end effector.

Instruments with 2 DOF at the tip may have a bigger shaft diameter (between 8 and 10 mm diameter) since they support more mechanical features in the internal structure such as rotation of the shaft and bending the distal tip or rotation of the distal tip and bending the distal tip. Even so, 5mm of diameter can be found on this kind of instruments. The distal tip bends up to 90º or even up to 180º in some instruments and when bending, the handle may move horizontal or vertical depending on the handle design.

Figure 10. Articulated laparoscopic instrument

Instruments with 3 DOF at the tip (Figure 11) integrate all movements above described. They have a rotating knob to move the shaft, the handle moves vertically or horizontally in order to deflect the distal tip, and another rotating knob that can be on the handle rotates the distal tip. The shaft diameter may be up to 10mm because more mechanisms are needed inside the device. An instrument with 3 DOF at the tip requires more hand movements than the other instruments with less DOF. Therefore, surgeons have to make much more effort reaching these mechanisms to control the device.

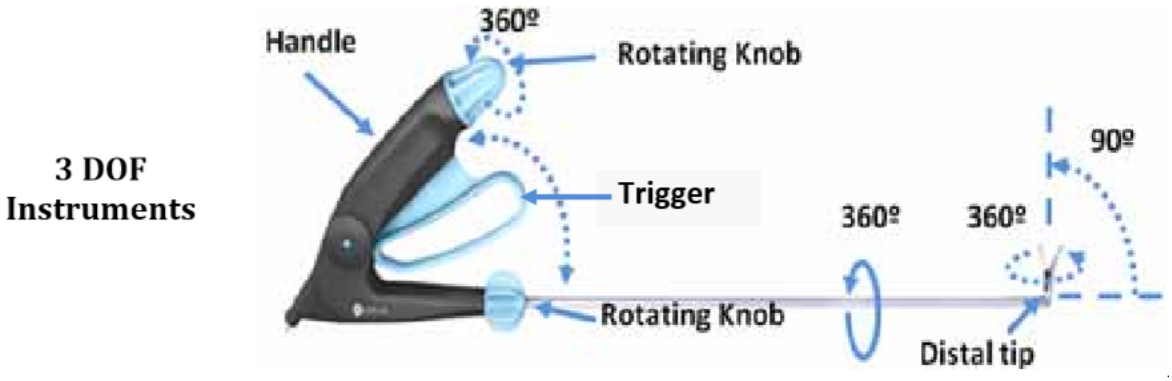


Figure 11. Articulated laparoscopic instrument of 3 DOF

4.2.3. Robotized surgical instruments

Robotics is increasingly used in laparoscopy, allowing surgeons to perform certain movements which are not possible with conventional laparoscopic instrumentation, thanks to the additional DOF at the instrument tip (Figure 12).

The additional DOF are moved by actuators, thus are controlled by an actuation system through electronic controllers manipulated by the surgeon. These instruments allow that one or more actuators placed in the internal part of the instrument can move and orientate the end effector.

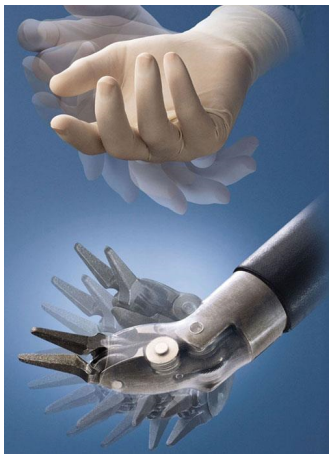


Figure 12. Robotized laparoscopic instrument

Usually the electronic controllers manipulated by the surgeon are very easy to use and provide high dexterity. The difference of the usage between mechatronic devices and conventional instruments is considerable. When manipulating mechatronic devices, any DOF can be controlled by means of joysticks, push buttons, etc., without much effort.

4.3. Teleoperated systems in robotic surgery

The concept of teleoperation was invented around 1940 and it was not related to surgery a human operator manipulates the master robot, and the slave robot copies the motion at the remote site. The master robot's position is sensed and a computer calculates the motor control signals needed for the slave to replicate the master's motion. There are bilateral systems where the slave has the ability to sense external forces and send this information to the control unit. By means of mechanical actuators, the resultant efforts are transferred to the master device for the surgeon sense a haptic feedback (Figure 13).

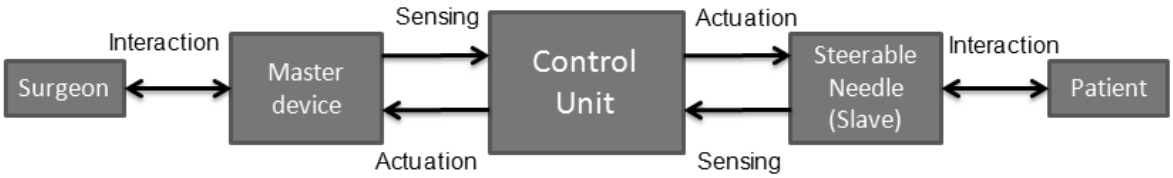


Figure 13. Diagram of teleoperated bilateral systems in robotic surgery.

One of the advantages of robotic surgery is that the surgical instrument can be replaced by instruments endowed with additional DOF. This allows surgeons to operate with better dexterity than in conventional surgeries.

In laparoscopy, these robotic instruments are controlled by the surgeons through teleoperation systems. The major advantage of these systems is the additional degrees of freedom available at the instruments' tip, which allow the surgeons perform more complex movements in a limited space.

Another important advantage of teleoperated robotics surgery is the option of scaling. The scaling down of surgeon's movements allows for surgical procedures to be more precise.

The unique teleoperated systems in the market is the *da Vinci Surgical System* (Figure 14). This system consists of a master-slave configuration. The surgeon operates from a console, which includes the master commands and the 3D imaging of the surgical scene. The scene captured by a stereo camera introduced into the patient through the laparoscope. By means of a set of pedals the surgeon can choose which surgical instrument is to be controlled with his hands.

This system can be separated in three parts:

- Surgeon's console. Is remotely located from the patient. The surgeon can control the robotic instruments via the master device and the system transfers the movements of the surgeon's hand, wrist and finger to the robot (slave). As mentioned above, the surgeon can also see a 3D image form the surgical scene by means of a 3D stereo

camera.

- Patient Side Cart. Contains the robotic arms and the surgical instrument, called endowrist. It is connected to the surgeon's console.
- InSite Vision System. High resolution 3D Imaging, which provide true sense of depth of the surgical procedure.

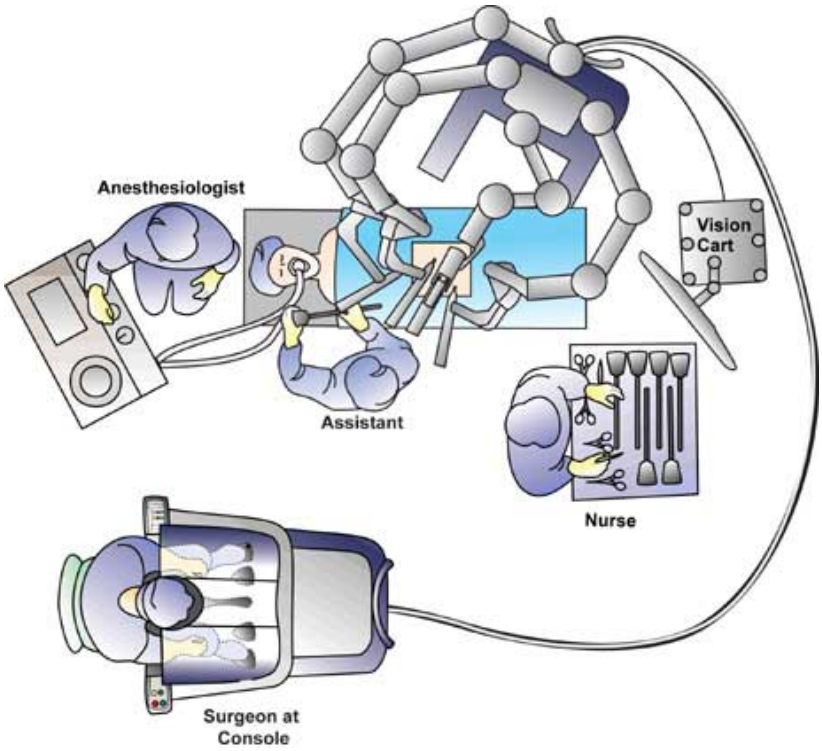


Figure 14. Da Vinci system.

5. WORK ENVIRONMENT

This chapter describes the work environment where the project has been implemented. The aim is that the reader acknowledges the different elements taking part of the system and their interaction, as well as they understand the overall working workflow. Starting with the description of the robotic system previous to the project, the necessary changes to implement our development will be presented afterwards. In the following chapter, the necessary design and implementation of these changes will be detailed.

The project takes place in the ESAII robotics laboratory, here six operating robot arms, are used for research in various fields. In our case, the working setup disposes of three robots, arranged in a way that laparoscopic surgical interventions can be simulated, though other configurations can be also possible (e.g. microsurgery and craniofacial surgery) (Figure 15).

Figure 15. Laparoscopy by robot arms.

The work environment has the essential elements to simulate a surgical theater, which are shown in Figure 16. Basically a pair of camera to monitor the surgical scenario, three robotic arms with their controllers, a couple of master haptic devices to teleoperate the robots and the user interface. For safety reasons, the motion of the master device will be transferred to the robots only when the user steps on the corresponding operation pedals.

Below, each of these elements is described in more detail:

Robotic arms: 3 robotic arms, branded Stäubli, which are endowed with 6 DOF each one. Two of these robots incorporate a surgical instrument of 1 DOF at the tip which is used to grasping. The third one holds the laparoscope.

Controller: Is the unit that controls the robotic arms movement. It has a Central Processing Unit or CPU. The CPU guides the robot through six numerical power amplifiers. This unit is

responsible for receiving commands sent via TCP/IP from the PC or by means of the manual control unit (MCP) and transmitting them to the robot so that it is run. Finally, the controller returns the corresponding message confirming that the order has been fulfilled.

Manual Control Pendant (MCP): It is a console that enables communication between the user and the CPU controller. It consists of a keyboard, LCD screen and button safety (emergency stop), ordered to block all actions should have an unwanted movement of the robot.

Video cameras: There is a set of cameras which are enabled for the teleoperation part. The video cameras have two main objectives in teleoperation. On the one hand, they provide visual information of the surgical scene to the surgeon. On the other hand, if more than one digital camera is used, several benefits can be obtained due to stereo vision(e.g.3D vision and depth information of the scene through image processing);

Haptic devices: Two *Phantom Omni Haptic* Devices with 6 DOF (6 rotational joints) are used as master devices to teleoperated one robot each. The user moves the tip of this device as if it was a surgical tool and, by means of the encoders at each joint, the 6 DOF can be computed. This information can be used, for instance to control the position and orientation of the robot end-effector.

Control board: Arduino board in charge of signaling the status of the pedals and the open/close status of the 1 DOF surgical instrument attached to the robot arms.

Pedals: Panel set of 4 configurable pedals. In the initial working setup, only two of them are linked to the robots holding the surgical instruments. They are used for safety reasons (the robots move only when their associated pedal is stepped on). When the scale factor between the movements of the master device and the robots is very high, the former tends to reach its physical limits. In this case, the pedals allow the master device to recover the central position without changing the robot's position.

Surgical instrument: The current surgical instrument is a conventional laparoscopic tool with one DOF at the tip. It is attached to the robot end effector together with an analogic servo that opens/closes the tip.

Figure 16 represents the configuration of the work environment set before the project. The user takes the control of the haptic device and the robots reproduce the user's movements with certain scale. The PC disposes of a software which captures the positions/orientations from the *Phantom Device*. The PC and the *Phantom* are connected via FireWire. The PC transform these data into commands for the robot controller, which immediately sends to the robots the signal and provides the power to do the desired movement. When the robot reaches the goal, the robot returns a confirmation message to the controller, and in turn, the controller

to the PC. In this manner a loop is generated for the robots to reproduce the master device movements. As it is shown in Figure 16 the control board is the link between pedals, PC and robots (including surgical instruments). Finally, the cameras can be connected to a monitor to visualize the scene or to the PC if some image processing is necessary.

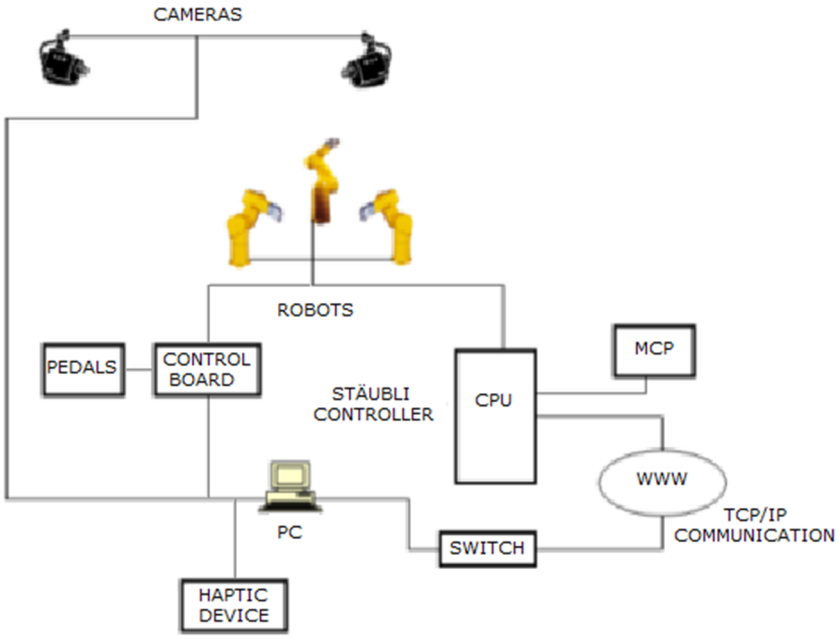


Figure 16. Work environment standard.

5.1. Working Setup

In order to test the new surgical tool, the current working setup has been modified (Figure 20). This section describes what changes have been done to teleoperate one robot, so that the tool's effectiveness is demonstrated. Although the system has been designed to control both operating robots the whole implementation is out of the scope of this project.

The main differences between the previous and the proposed working setup are located in the PC software, the control board and, obviously, the surgical tool.

The **software's interface** reads the position and orientation data of the *Phantom Haptic Device*. The translation increments of the *Phantom's* tool are computed and translated into scaled incremental positions of the robot. The absolute orientation of the phantom is also captured by the PC and sent to the control board in form of three angles.

The **control board** has been updated in order to control all the servos incorporated at the new surgical instrument. This update comes with a new communication protocol to transfer the tip orientation from the haptic device (through the software PC) to the tool's tip and also to get the pedals status. The board transforms these angles into rotations of the four servos, making the

surgical instrument's tip replicate the phantom's orientation.

It has been taken into account that the coordinate systems of the robot and of the phantom are different. Therefore, the difference between axes orientations is considered when sending both the new position of the robot and orientation of the tool's tip.

The laparoscopic instrument selected for this project is an *Intuitive Surgical's EndoWrist® Forceps* (Figure 17) with S system (Figure 18) which is used in *da Vinci* robot (Figure 19). However, our design will be valid for any surgical instrument of *da Vinci* robot. This tool is a consumable instrument of the *da Vinci* robot from Intuitive Surgical Inc.

Figure 17. Intuitive Surgical's EndoWrist® Forceps.
a) Back part. Pulley system.
b) Tip of the forceps.

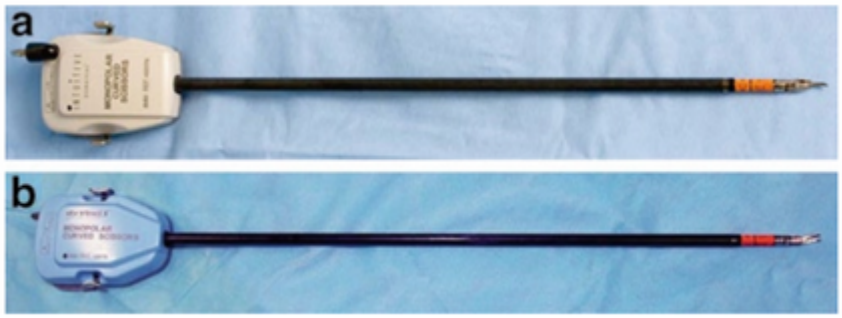


Figure 18. EndoWrist® instrument for the standard (a) and S (b) systems.

As it can be seen in the Figure 17.a this robotic instrument is a pulley passive system part. The transmission system to move the tool tip requires four actuators, the choice which is critical for both mechanical and electronic designs described in the chapter 6.

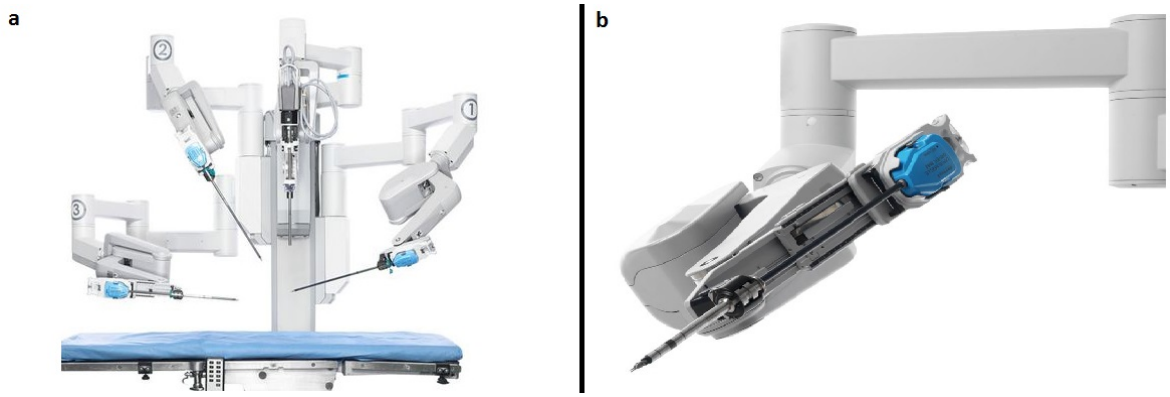


Figure 19. a) Intuitive Surgical's da Vinci Robot.
b) Instrument replacement system.

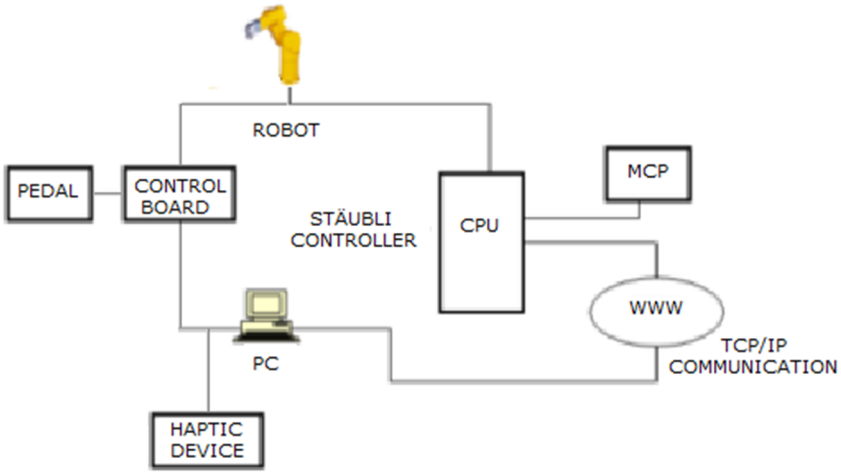


Figure 20. Working setup.

5.2. Expected Technical Specifications

To evaluate the behavior of the new surgical instrument it would have been interesting to benchmark the current 1 DOF tool, but unfortunately no features or protocols to compare the dexterity of both instruments are available. Accuracy, execution time and economy of motion are some typical parameters to assess the quality of a surgical task. In our case, however, they are not relevant, since the new surgical instrument does not bring any improvement with respect to the old instrument in those tasks that both instruments can perform. The key point is that the use of the conventional surgical instrument used up to date is limited only to basic exercises. For example, in order to avoid unexpected movements of the robot, two DOF of the robot were fixed in surgical applications previously done in the laboratory, corresponding to pitch and yaw orientations for the robot's end effector. The dexterity comparison between the instruments is expressed by the increment of available orientations of the tool, given a regular working position.

To demonstrate the effectiveness of the system's dexterity, some trials of a suturing tasks have been executed, which the robot was not able to perform with the conventional surgical instruments.

A 3D model of the robot and a software simulator connected to the robot's server have been used to show the orientation limits of the conventional surgical instrument. The robot model starts in a regular position suitable for surgical intervention (Figure 21) and it is controlled by a 3D mouse.

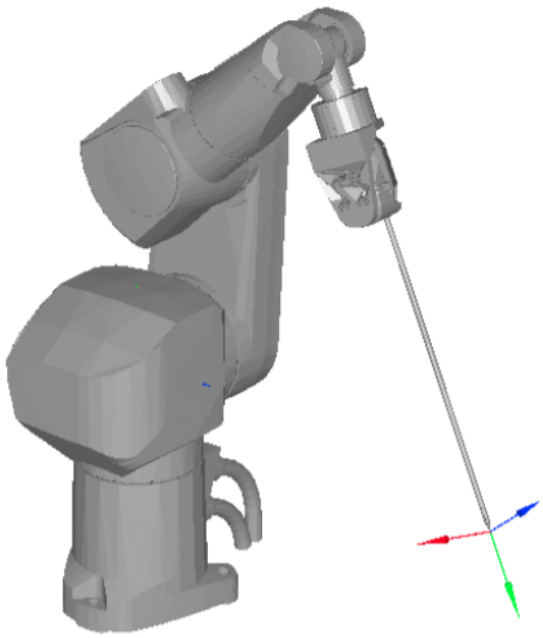


Figure 21. Initial pose.

The simulator allows fixing the position of the tool tip and changing its orientations with respect of the three axes. The controller computes the robot's configuration through reverse kinematics and the simulator displays it. In the case that any joint limit is reached, the simulator does not allow moving the robot further. The angles of the tool tip in these limit positions are registered, so that the ranges of available orientations can be determined.

The orientation of the tool's tip are represented in Figure 21 by the X, Y, Z axes, colored in blue, red and green respectively. The rotations of these axes are expressed in angles alpha (around X), beta (around Y) and gamma (around Z), also known as alpha, beta and gamma orientations.

The instrument's shaft is aligned with the end effector of the robot, so the limits of the gamma orientation coincide with the limits of the robot's 6th joint, which can rotate 540º. In this case, no simulation is needed.

In Figure 22, the limits of the alpha orientation are shown. The robots are displayed in their limit positions, having rotated 116º the tool tip around the Y axe. Although that range is not very limited (116º), robot's joint configuration has suffered large displacements (e.g. joint 1 has suffered rotated almost 90º).

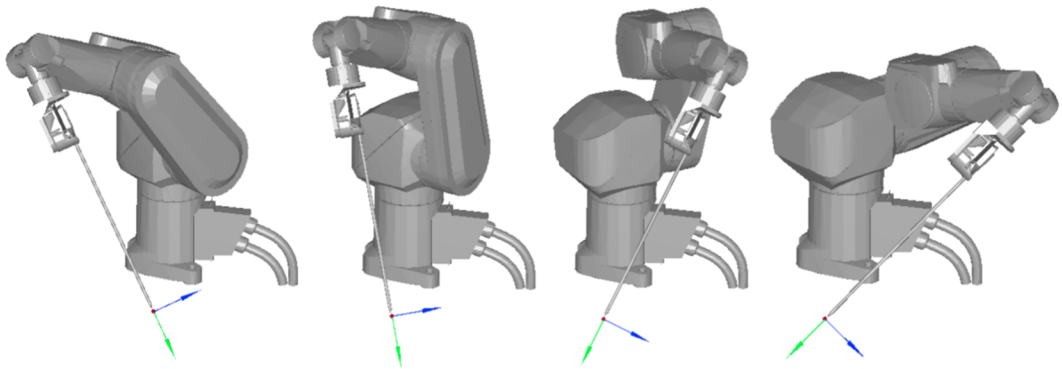


Figure 22. Robot's pose in the alpha limits [144º, 118º, 45º, 28º]. View of plane YZ.

The same process has been followed to calculate the beta orientation range (Figure 23). As a result, is the tool tip has been able to rotate only 35º. As in the case of the alpha orientation, the displacements of the joints are considerable. Note, in Figure 23 between the 1st and 2nd frame, the configuration of the robot is totally different and the increment acquired is of 1º.

With the contribution of this project, it is expected that the robot improves the limitation of low orientations range and requires shorter joints displacements.

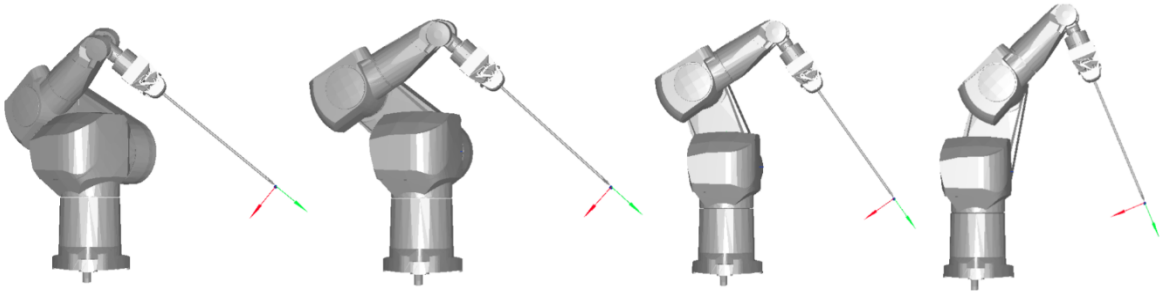


Figure 23. Robot's pose in the beta limits [130º, 131º, 141º, 165º] View of plane XZ.

The following table show the summary of the limits and ranges of each orientation.

Table 1. Limits and range orientation at the tip's tool.

	Lowe limit angle	Upper limit angle	Angle range
Alpha	28	144	116
Beta	130	165	35
Gamma	-270	270	540

6. METHODOLOGY

This chapter refers the strategy followed during the project's development. The whole work can be divided in several parts, which are chronologically described below.

The first step consisted of analyzing the new surgical instrument. It is essential to understand the behavior of the tool in order to develop a good coupling. The tool was originally designed to work with an existing robotic arm, thus its design comes from the necessities of this robot and its applications. For this reason the coupling parts to be designed in the project must fit this tool properly in order to extract its full potential. Since the new instrument is motorized, the first constrain is given by the choice of the actuators in charge of moving the tool tip. The first specifications to be defined are directly related to the geometry of the instrument coupling and the actuators' power.

Once the geometry and the power of the actuators' power are evaluated, the mechanical parts are designed to be manufactured by a 3D printer. This technology reduce the production costs, from the prototyping phase until the final design is achieved. The actuators will not be designed in house, but acquired from the market after comparing the commercially available models.

The self-engineering contribution of the project is dived into the following phases:

- **Mechanical design**: of the coupling and transmission between the actuators and the tool plus the coupling between the tool and the robot.
- **Electronic design**: of the control board which control the pedals status and the positioning of the actuators.
- **Communication**:
    - Protocol development to establish communication among the control board the PC and the actuators.
    - Software: development for the interfaces among the user, the master device and the robot.
- **Manufacture and assembly** of the designed parts.

Finally, some tests have been done to check the effectiveness of the new system, as described in chapter 7. EXPERIMENTAL DESIGN, where the analysis of the results will be also explained.

6.1. Mechanical Design

In this section, the mechanical parts required for the coupling of the new surgical instrument are described. The mechanical design comprises the actuators selection, the design of the transmission between the actuators and the instrument and also the design of the mechanical couplings. The latter, includes a coupling for the actuators in the instrument and the coupling for the whole instrument (it includes the forceps with the motor assembled).

6.1.1. Actuators selection

The motion of the surgical instrument is provided by four actuators, the specifications of which have to fit several requirements, such as power, size, available features and price. This section describes the criteria followed to choose the proper actuators.

According to the nature of the surgical task, the gasping force of the tool tip requires a torque of **0.2 N·m**. This maximum torque is defined by the maximum force that can withstand the pulleys at the tip of the tool. Provided that the force transmission ratio of the pulleys is 1:1, the torque specifications of the chosen servos must be over this value.

In the robotic industry there are mainly two kinds of servos, the digital servos and the analog servos. The current surgical instrument of 1 DOF installed in our robots is using an analog servo to open and close de gripper. The new surgical instruments does not only control the opening and closing of the gripper, but it is endowed with 3 DOF for pitch roll an yaw, including opening and closing of the gripper.

A digital servo includes a small microprocessor inside which analyzes the received signals and processes them into high frequency voltage pulses to the servo motor. Instead of 50 pulses per second used in an analog servo, the digital servo motor receives up to 300 pulses per second. The pulses are shorter in length, but the large number of voltage pulses speed up the motor much quicker and provide a constant torque. As a result the servo has a much smaller deadband, a faster response, a quicker and smoother acceleration, and a better holding power [7].

Additionally, using a digital servo motor, the small microcontroller may have the ability of defining the maximum torque that can be applied. This parameter is used to avoid breaking mechanical parts and also to get internal information, as the temperature of the motor.

A well-recognized digital servo is made by ROBOTIS [8] branded DYNAMIXEL (DXL). From their catalogue, the digital servo XL-320 (Figure 24) was selected, with a stall torque of 0.39 N·m at 7.4V which and a resolution of complies with the specifications requirements, the resolution of which is 0.29 degrees.

The model XL-320 is the smallest digital servo from DXL. The next upper model is the AX-12A with a stall torque of 1.5 N·m at 12V. Both servos fit the technical requirements, hence the chosen model has been th XL-320 due to its lower cost.

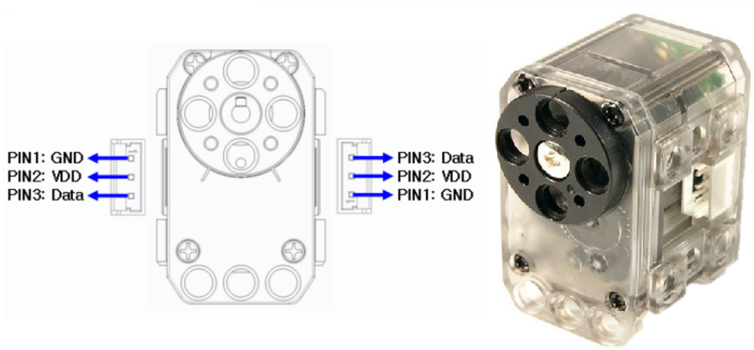


Figure 24. XL-320 DYNAMIXEL model from ROBOTIS.

The XL-320 model specifications are summarized below:

- Weight : 16.7g
- Dimension : 24mm *36mm * 27mm
- Resolution : 0.29°
- Motor : Cored Motor
- Gear Reduction Ratio : 238 : 1
- Stall Torque : 0.39 N·m (at 7.4V)
- No load speed : 114 rpm (at 7.4V)
- Running Degree
    - o 0° ~ 300°
    - o Endless Turn
- Running Temperature : -5°C ~ +70°C
- Voltage : 6 ~ 8.4V (Recommended Voltage 7.4V)
- Command Signal : Digital Packet
- Protocol Type : Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
- Link (Physical) : TTL Level Multi Drop (daisy chain type Connector)
- ID : 253 ID (0~252)
- Communication Speed : 7343bps ~ 1 Mbps
- Feedback: Position, Temperature, Load, Input Voltage, etc.
- Material : Engineering Plastic

6.1.2. Transmission

Once the servos have been chosen, the information needed to design the transmission can be obtained by linking the geometries of the servo with the transmission pulleys of the surgical instrument (Figure 25).

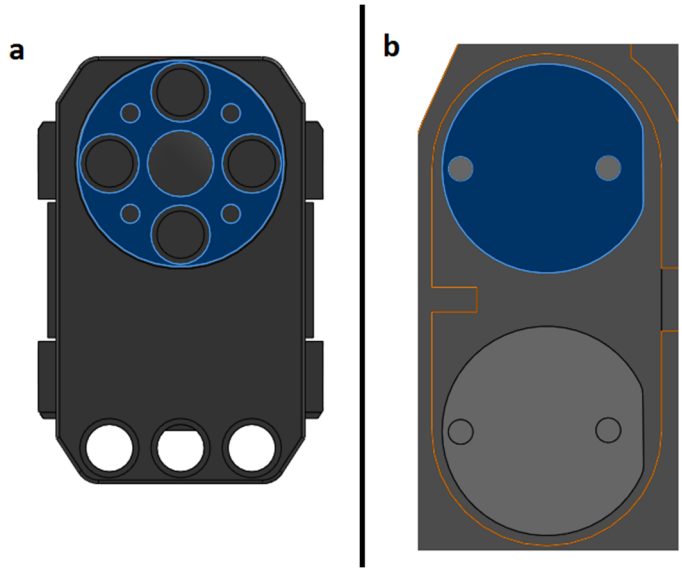


Figure 25. Transmission interaction.
a) . XL-320 DYNAMIXEL.
b) Passive forceps pulley system

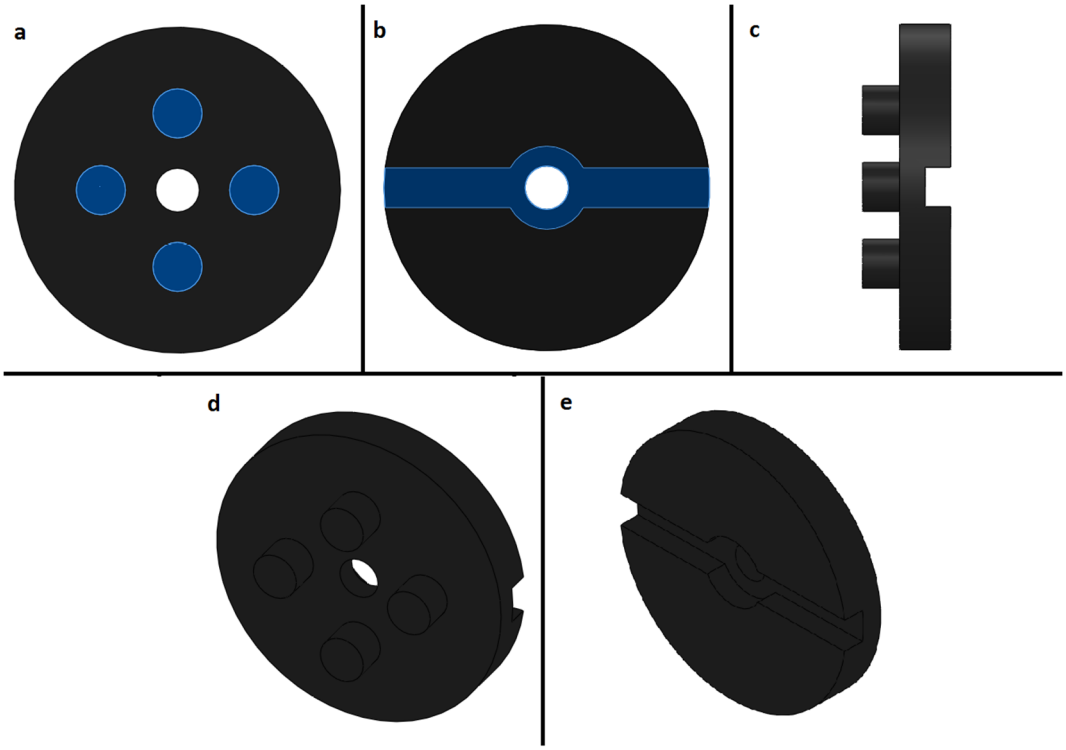


Figure 26. Transmission part.
a) Transmission servo face. The blue zones go into holes of the servo shaft part.
b) Transmission forceps' pulley face. The blue zone is the socket for the pivots of the forceps' pulley.
c) Profile view.
d) Isometric view of the transmission servo face.
e) Isometric view for the transmission forceps' pulley face.

The transmission system can be seen Figure 26. It contains 4 pivots in the servos side (Figure 26.a) which fit the holes of the servo shaft part, so the transmission part can be well assembled

to the servo. On the other side (Figure 26.b) the pulley transmission has a socket for the pivots of the instruments' pulley.

Figure 27 shows the transmission assembly of the servo shaft and the forceps' pulley through the transmission part.

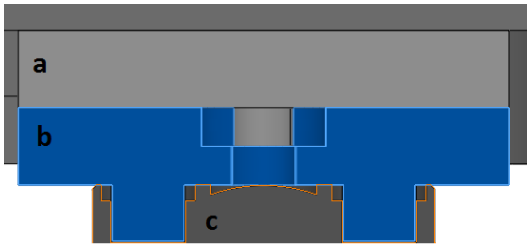


Figure 27. Transmission assembly.
a) Forceps' pulley.
b) Transmission part.
c) Servo shaft.

In Figure 28 the final assembly of the four servos with the forceps' pulley system is shown.

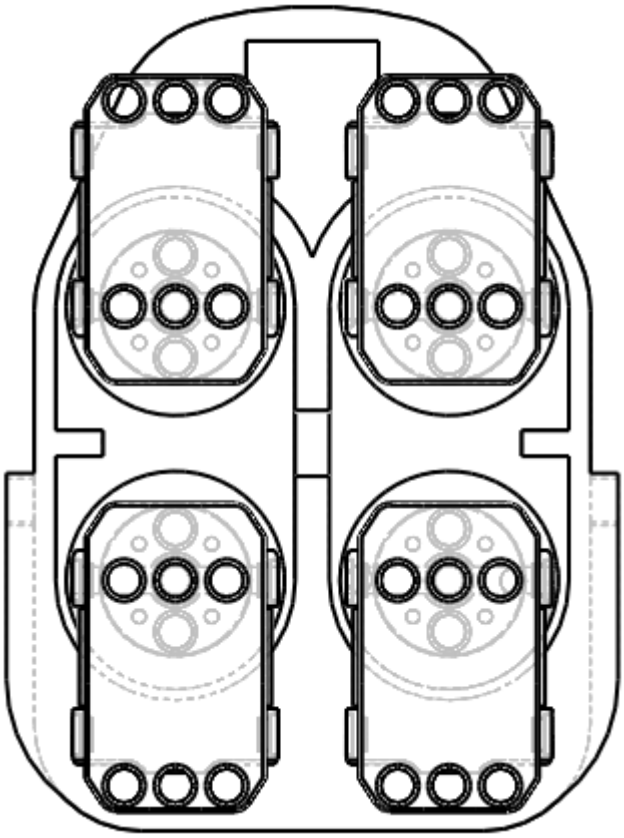


Figure 28.Servo, transmission and forceps' pulley system assembly.

6.1.3. Mechanical couplings

The most complex task of the mechanic design has been the mechanical coupling, due to the complexity of the tool geometry. Several iterations were needed until they got to have a good fit between the coupling and the tool. As in the case of the transmission parts, all the parts have been created through Fused Deposition Modeling technique (FDM).

The coupling is divided into two parts. The first part is the coupling between the servos and the instrument's pulley system and the second one is the part responsible for assembling the whole instrument with the robot.

The design has started with a base for all the components to be included. This base is shown in yellow in Figure 29, and it is coupled to the instrument by a fixing support, shown in green.

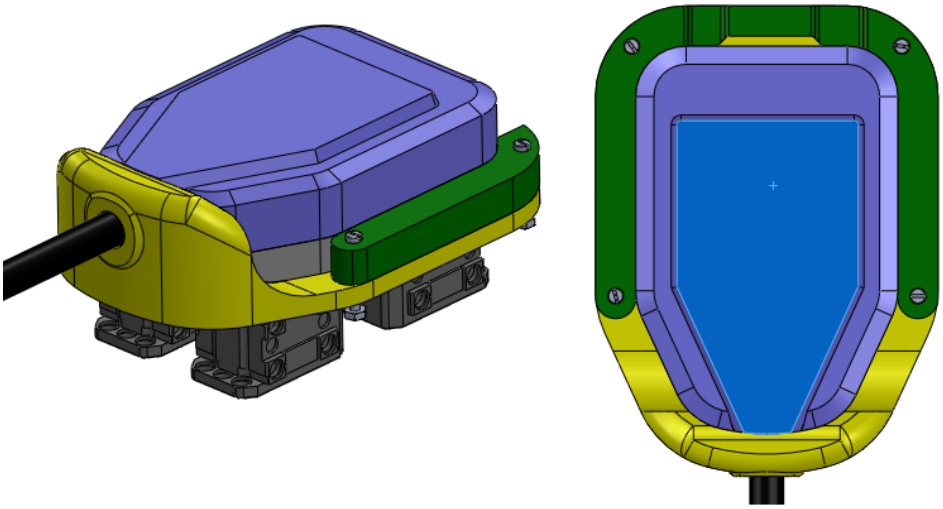


Figure 29. First coupling level.

As Figure 29 shows, the base (in yellow) has been designed also to reduce the vibration of the 460mm long instrument's shaft, by means of a hole with a bearing. The bearing shown in (Figure 30) allow keeping the roll DOF smooth. The green piece fully fixes the base with the instrument.

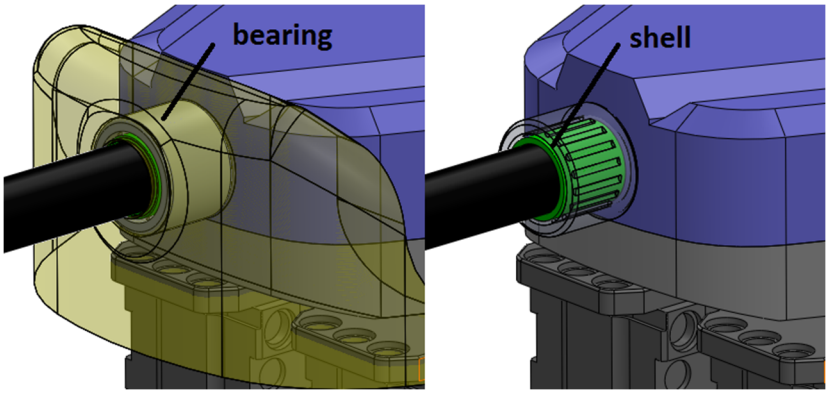


Figure 30. Detail of the fixing part of the rod.

The Figure 31 shows the housing servos (red part). To fix this part, some screws fix the yellow and green parts, minimizing then the number of screws used to assembly the servo's part.

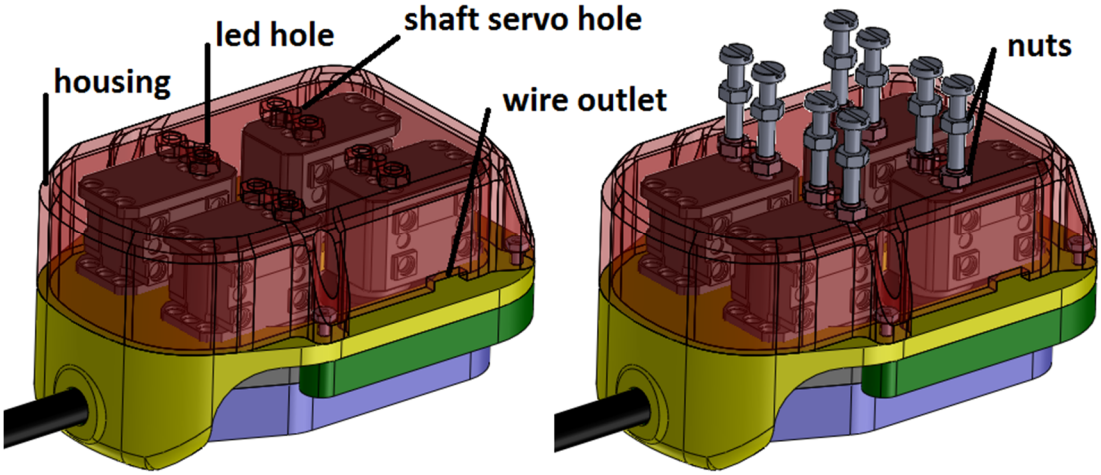


Figure 31. Housing servos.

The XL-320 servos are provided with a RGB LED. This LED is used to indicate several working status (waiting, running, error), which can be useful for diagnosis purposes in case of failures. This LED can be seen through a hole in the housing of the servos. Four additional holes allow the manual movement of the servos' shafts, hence the manual movements of the instrument. The housing also incorporates a wire outlet for communication and power supply.

The fixing holes of XL-320 are very fragile being the stronger ones the fixing holes located at the bottom. For this reason, these have been the fixing holes chosen to fix the servo to the structure. As Figure 31 left shows, each servo is fixed to the housing with a M4 metrics screw, and it has been needed to embed nuts in the housing (red part in Figure 31 right). The screw used for this part has a length of 30mm and is assembled with an extra nut which is used to fix it to the coupling robot-instrument showed in Figure 32.

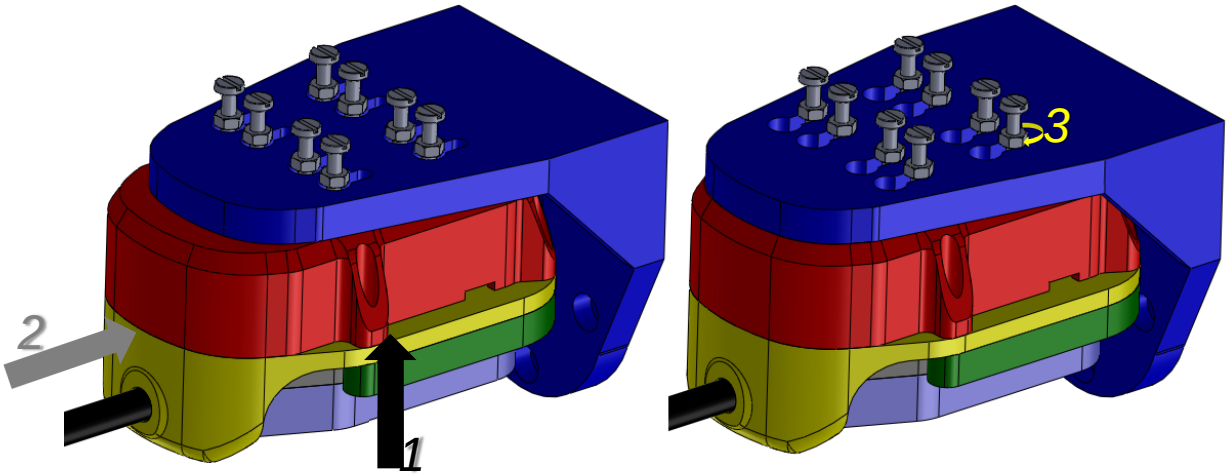


Figure 32. Coupling system to the robot.

As the Figure 32 shows, only three steps are needed to assemble the whole instrument to the robot. The blue part becomes solidary to the link 6 of the robot where the new instrument is assembled (Figure 33).

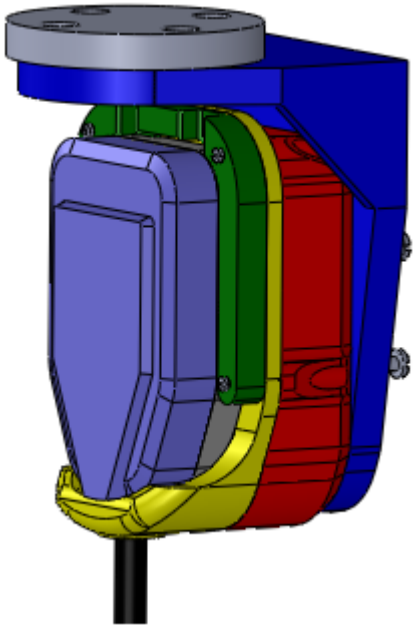


Figure 33. The coupling system for the robot is assembled to the link 6 of the robot.

6.2. Electronic Design

In this section the electronic part included in the project is explained. This control board will be connected to the PC via USB to transfer the data in both directions. The PC will send a request to the control board asking for the pedal status. The Arduino will answer this request. If the pedal status indicates that the pedal is pressed, the PC will transfer the orientation read by the haptic device to the control board. The control board should interpret the data received and prepared it to send it in the properly manner to the servos.

6.2.1. Components Selection

As explained in sections above, the instrument chosen for this project use a pulley system which endows their tip with 3 DOF. The system also provides the grasping action by moving each pulley with the servos (Figure 24).

The whole mechanical system actuated by four DYNAMIXEL XL-320 servos. Theses servos use a custom bus protocol vial Half Duplex Asynchronous Serial Communication (HD-ASC) of 3 wires (1. GND, 2. VDD, 3. DATA) as the Table 2 shows. This kind of communication allows connecting 253 servos in the same bus.

Table 2. Dynamixel model-specific connector's standard for XL-Series.

XL-Series	
	
1	GND
2	VDD
3	DATA

In order to drive the servos, an electronic circuit has been designed, including additional components the electronic circuit needed in order to implement the HD-ASC. The integrated circuit used is the SN74LS241N.

The electronic board requires a microcontroller to receive the data from PC, send the desired actions to servos, and return the state of the pedals and the servos to the PC.

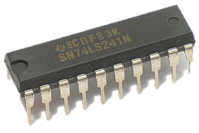
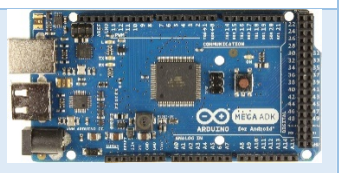
The selected one has been an Arduino, because it provide to our system the functionalities needed and it is easy to program and to implement in the prototype.

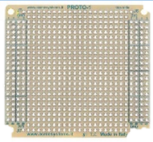
The Arduino model selected must have almost more than one set of serial communication port since the XL-320 servo uses a serial communication port to receive and send data to the microcontroller to the PC and also for the PC-Arduino communication using a USB.

In this case, the most common model of Arduino is the Arduino Mega. Moreover it has 4 serial ports one of which is dedicated to USB communications, which is perfect for this project.

The Table 3 shows the components used in the electronic circuit. The overall cost is exposed in the Project cost section included in the ANNEXES.

Table 3. Electronic components' list.

Component	Units	Description	Image
DYNAMIXEL XL-320	8 4 units by tool (2 forceps)	The DYNAMIXEL XL-320 is a robot smart actuator with a fully integrated DC Motor + Reduction Gearhead + Controller + Driver + Network in one small DC servo module.	
Connectors	4 1 unit by each robot and 1 unit for each pedal (2 robots and 2 pedals)	Connector of 3 wires for PCB	
Pedals	2 1 unit by each robot (2 robots)	The pedals are an electronic device used to get or take movement to robots and forceps.	
SN74LS241N	1	SN74LS241N is an octa buffer commonly used for HD.ASC for DYNAMIXEL servos.	
Arduino Mega 2560	1	This is a microcontroller used in order to integrate and communicate the electronic part	

		with the motorized instrument. This microcontroller processes the information coming from PC.	
little proto PCB of holes	1	This is the base to put all electronic parts in an organized way.	

6.2.2. Electronic Layout

This subsection describes the final electronic layout. The electronic layout takes into account all parts described in the above subsection and it is shown in Figure 34. The electronic layout of this project represents the Arduino shield for Arduino Mega 2560. It contains the input and output connectors for the servos and the pedals respectively to control two robots. It also includes the integrated circuit SN74LS241N, which communicates universal asynchronous receiver/transmitter (UART) with a HD-ASC as the Figure 35. The Figure 36 shows the scheme of the SN74LS241N used for a HD-ASC.

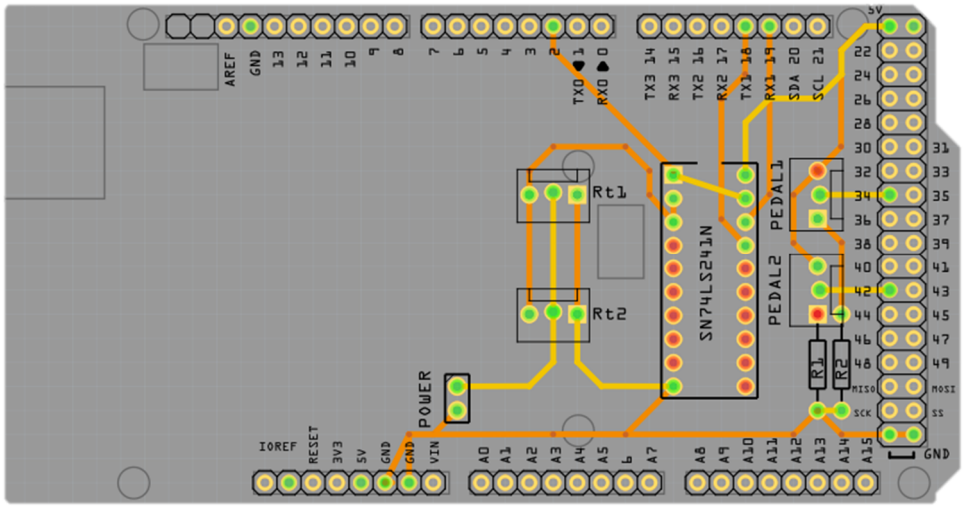


Figure 34. Arduino shield.

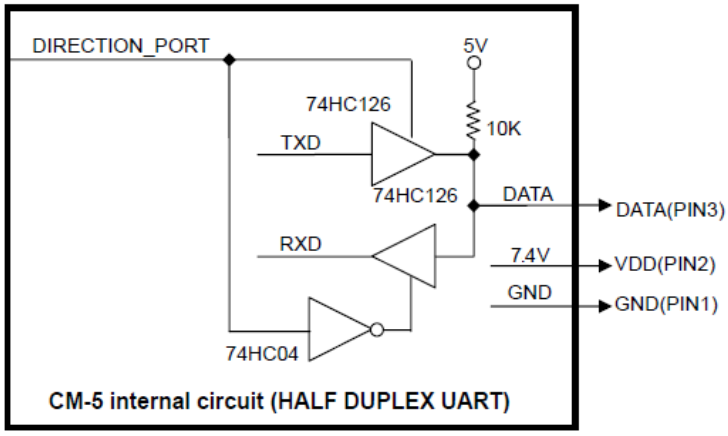


Figure 35. Half Duplex UART.

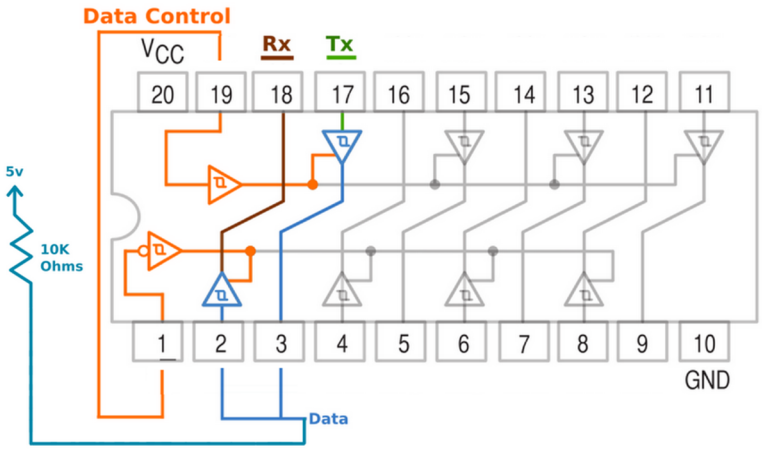


Figure 36.SN74LS241N diagram for HD-ASC UART.

6.3. Control & communications

In this section, the control systems and the implemented communication protocols are described. This part explains how the servos are controlled via a customized communication protocol. A brief explanation of how the *Phantom Omni Haptic* Device, the robots and the PC works and how all this is implemented in the present project is also included. Finally a whole system will be presented as a summary of all this section.

6.3.1. Control and communication of the XL-320 servo

The position of the servos is controlled by a microcontroller inside. This microcontroller controller the position of the DYNAMIXEL sevo through a PID controller as it is shown in the Figure 37. The values of the PID gains can be modified through DYNAMIXEL communication protocol.

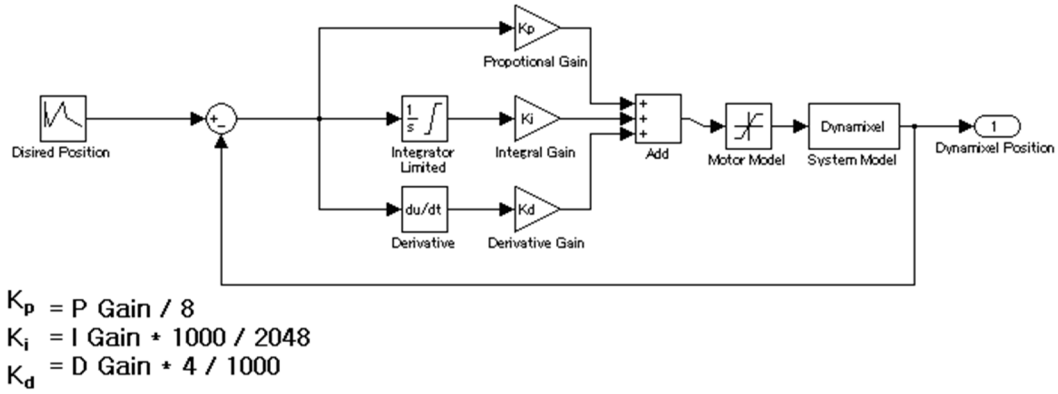


Figure 37. XL-320 PID

The behavior of the motor has a compliance to set the control flexibility of the motor. The Figure 38 right shows the relationship between the output torque and the position of the motor. The Figure 38 left shows the limits of the motor when it works in absolute angles (0-300).

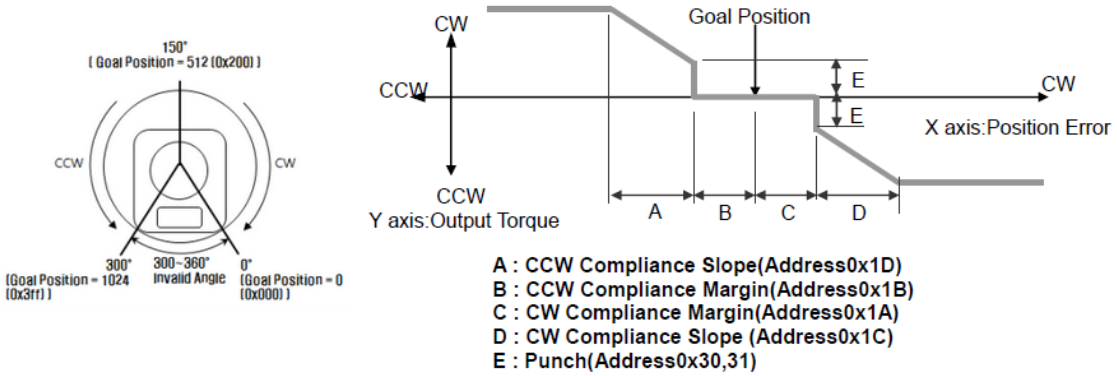


Figure 38. XL-320 diagram

In order to control the position of the servo, the Arduino sends the desired position and the servo's microcontroller converts this signal into the voltage needed to move the servo's shaft in the desired position. In the case the servo response is not good enough, the PID gains should be changed.

The second part in this subsection is how to communicate the desired action to the servo.

The communication protocol used by the servos is provided by the manufacturer. Table 4 shows the structure of this protocol.

Table 4. ROBOTIS protocol for DYNAMIXEL 2.0 version [9].

0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	INST	Param1	…	ParamN	CRC_L	CRC_H
Header			Reserved	ID	Packet Length		Instruction	Parameter			16bit CRC	

A. Header : Field indicating start of packet
B. Reserved : 0x00 (cannot use 0xFD)
C. ID : Dynamixel ID designated for Instruction Packet processing
    1. Value range: 0 ~ 252 (0x00 ~ 0xFC) 253 unique values
    2. 253(0xFD), 255(0xFF) in Header prohibited use.
    3. 254 (0xFE): Broadcast ID, ALL connected Dynamixels controlled by Instruction Packet.
D. Packet Length : Packet length after Packet Length field
    1. [Instruction] [Parameter] [CRC_L] [CRC_H], Parameter length + 3.
E. Instruction: individual packet's command field
F. Parameter: when auxiliary data is required. Depends on the instruction.
G. 16bit CRC: CRC-16 values to verify reliability of data.
    1. CRC-16 values can be used against packet communications to check for damages on data.

The implementation of the protocol shown above in the Arduino requires a library in order to establish communication with the servo. Although many libraries can be found by internet, well tested and ready to use, they are not updated to the newest protocol version 2.0 for Arduino boards. Some libraries have been found, but their implementation does not match our needs.

Some of them can only send, but not receive data. Other use the half-duplex system by software instead of by hardware. This part is important for this project, because send and receive data is needed and this communication should be as fast as possible.

So, for this project, a library has been created to communicate the servos with the board through the protocol 2.0 version. This one is based in a previous library created by *Savage Electronics* [10] which is free software and it can be redistributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation. This library was designed for the protocol 1.0 and this was full and well implemented, using half-duplex hardware to send and receive data properly. Although this

library does not contain all the features provided by the XL-320 servo, it fully satisfies the features required to control the servos, therefore the tool's orientation. All the specific information related to the modified library is attached to the ANNEXES (section XL-320 Arduino library)

On the other hand, a short protocol has been designed to perform the communication between the PC and the Arduino. The data that the PC sends to the Arduino is the desired servo to be moved, the desired position of the servo, the desired maximum torque of the servo and the desired maximum speed. This protocol has the following rules:

1- The parameters' order is floating, but the position's parameter should be the last one.
2- Servo nomination → I###, where ### must be an integer from 0 to 254 which indicates the servo' ID.
3- Maximum torque → T###; where ### must be an integer from 0 to 100 which indicates the percentage of the maximum torque to reach the goal.
4- Maximum speed → S###, where ### must be an integer from 0 to 100 which indicates the percentage of the maximum speed to reach the goal.
5- Position desired → P###, where ### must be an integer from 0 to 300 which indicates the angle position. This parameter is the only one that generates movement to the servos and for this reason should be the last one.
6- To close the frame, it must be finished by ';'

Example:

Single movement

I1 T100 S100 P260; → Servo with ID 1 (I1) move at position 260º (P260) with maximum % torque (T100) at maximum % speed (S100).

Multi movement

I0 T50 S100 P20 I1 T100 S100 P260 I2 T100 S100 P200 I3 T50 S50 P0; → Servo with ID 0 (I0) move at position 20º (P20) with a torque of 50% (T50) at maximum speed (S100), Servo with ID 1 (I1) move at position 260º (P260) with maximum torque (T100) at maximum speed (S100), Servo with ID 2 (I2) move at position 200º (P200) with maximum torque (T100) at maximum speed (S100) and Servo with ID 3 (I3) move at position 260º (P0) with a torque of 50% (T50) at speed of 50% (S50).

I254 T100 S100 P180; → move at position 180º (P180) with maximum torque (T100) at maximum speed (S100) all servos (I254).

This protocol allows requesting the pedals state by just sending an 'X' and the Arduino will

answer with the pedal's state in the following way:

- 00 Pedal 1 not pressed, pedal 2 not pressed.
- 10 Pedal 1 pressed, pedal 2 not pressed.
- 01 Pedal 1 not pressed, pedal 2 pressed.
- 11 Pedal 1 pressed, pedal 2 pressed.

6.3.2. *Control and communication of Phantom Haptic Device*

Haptic feedback has been used in human robot interaction to provide the signals intended for the proprioceptors of the operator, i.e., force and position. Such especially dexterous manipulation of objects using hands or tools depends heavily on the tactile feedback.

As it is shown in the Figure 39 and Figure 40, the *Phantom Omni Joystick* is a 6 axis joystick. Joints 1 to 3 are used for position and joints 4 to 6 for orientation. The Table 5 shows the specification of the *Phantom Omni device.*

Figure 39. Position joints of Phantom Omni.
Joints 1 to 3 are used to capture the position's device

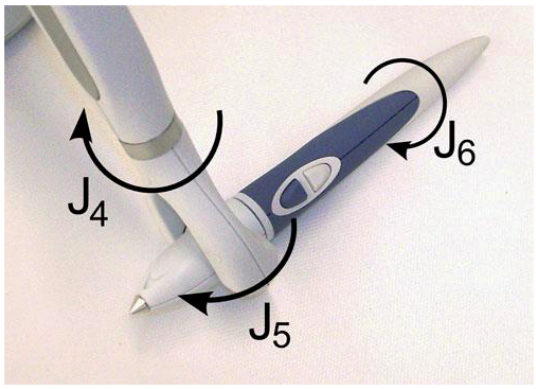


Figure 40. Inclination joints of Phantom Omni.
Joints 4 to 6 are used to capture the inclination's device

The *Phantom Omni Joystick* is connected to PC via IEEE 1394 Interface, which is an interface

standard for a serial bus for high-speed communications and isochronous real-time I/O data transfer. It was developed in the late 1980s and early 1990s by Apple, which called it FireWire [11].

Table 5. PHANTOM® Omni™ Force Feedback Joystick specifications [12].

Force feedback workspace	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Footprint (Physical area device base occupies on desk)	6 5/8 W x 8 D in ~168 W x 203 D mm
Weight (device only)	3 lbs 15 oz
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 450 dpi ~ 0.055 mm
Backdrive friction	< 1 oz (0.26 N)
Maximum exertable force at nominal (orthogonal arms) position	0.75 lbf (3.3 N)
Continuous exertable force (24 hrs)	> 0.2 lbf (0.88 N)
Stiffness	X axis > 7.3 lbs / in (1.26 N / mm) Y axis > 13.4 lbs / in (2.31 N / mm) Z axis > 5.9 lbs / in (1.02 N / mm)
Inertia (apparent mass at tip)	~0.101 lbm (45 g)
Force feedback	x, y, z
Position sensing [Stylus gimbal]	x, y, z (digital encoders) [Pitch, roll, yaw (± 5% linearity potentiometers)
Interface	IEEE-1394 FireWire® port: 6-pin to 6-pin
Supported platforms	Intel or AMD-based PCs
OpenHaptics®Toolkit compatibility	Yes
Applications	Selected Types of Haptic Research, FreeForm®Modeling™system, ClayTools™ system

In order to obtain information from these devices, the software environment used in the project has been Visual Basic with VC++ 2010 language. In order to control the devices, PHANTOM Device Drivers (PDD) need to installed, as well as OpenHaptics Toolkit library.

OpenHaptic is divided into several layers as it is shown in Figure 41. The levels of abstraction of the OpenHaptic Micro API For this project HDAPI (low level) and the HLAPI (high level) are combined.

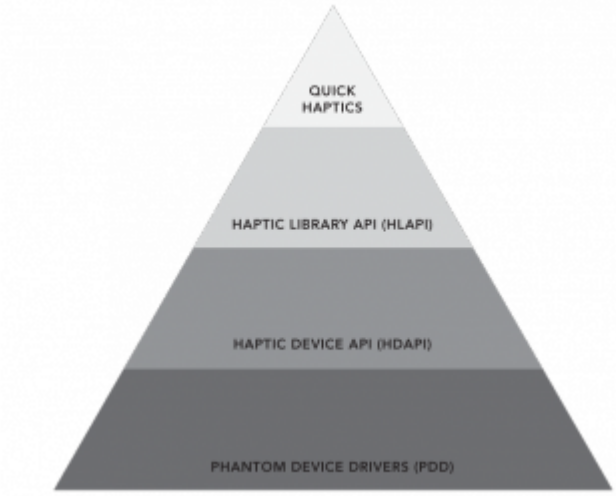


Figure 41. The levels of abstraction of the OpenHaptic Micro API [13]

The PC works as a bridge between the robot and the Phantom Omni, merging the robot framework and the OpenHaptic API.

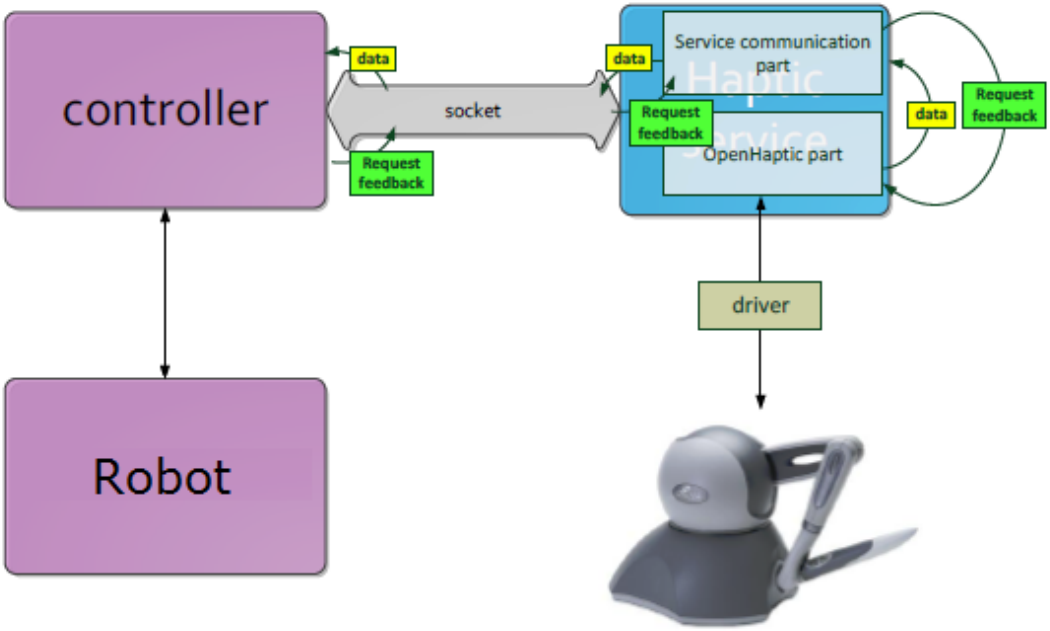


Figure 42. The Scheme of the communication between the phantom Omni and the controller [14].

The PC enables communication with a controller using sockets. The role of the service is to capture movement, the orientation and the state of buttons of the phantom Omni. Then the data is sent as a string to the controller (yellow part in Figure 42).

The green tag in the Figure 42 is the data that could come from robot in order to have a feedback force, in case that force sensors were used. In this project, though, force feedback is not implemented.

The structure of the data consists of 3 variables type floats for the position, 4 floats for the orientation and 1 integer for the buttons' state.

The PC program uses a thread for the Phantom, a loop separated from the main routine, which captures all these data in each loop and sends them scaled to the robot provided if the pedal is pressed. For more information see the Phantom Omni Joystick Thread section in the annexes.

6.3.3. STÄUBLI Robot

As explained in the Working Setup section, our lab disposes of 3 STÄUBLI robots RX60B (Figure 43), which are used for research purposes, mainly in the field of robotic surgery. The RX60B robots have 6 DOF and can lift up to a load of 2.5 kg.

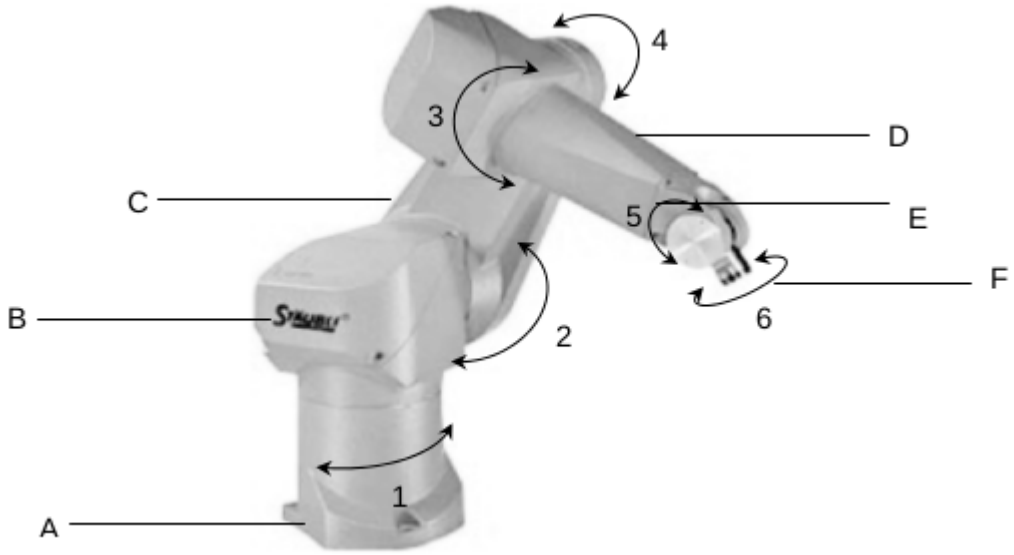


Figure 43. Robot Stäubli (RX Series 60B).
1-6 joints.
Base (A), shoulder (B), arm (C), elbow (D), forearm (E) wrist (F).

The RX60B robot is controlled by the STÄUBLI controller CS8. The controller is connected directly to the robot and is also connected to the PC via Ethernet. This controller incorporates a Manual Control Pendant (MCP) capable to run programs for the robots, to display the position and orientation of the joints and the end effector, as well as to move them manually through its keyboard (Figure 44).

For this project a code called INTERPRET is used. This code is programed in VAL3 language and it is used to translate the PC data into commands for the robot. The INTERPRET returns the robot's position feedback, received from the controller, once the action when it is done. The communication between the PC and the controller is via TCP/IP.

Figure 44. Manual Control Pendant (MCP)

6.3.4. Communications layout

As a summary of all the previous information, Figure 45 shows how the system is communicating with the other parts of the subsystem.

It can be noticed that the PC is the main controller, since it is connected to all the parts of the system. The PC receives the data from the haptic device and the PC distributes the data to the robot controller and the instrument's controller (Arduino Mega).

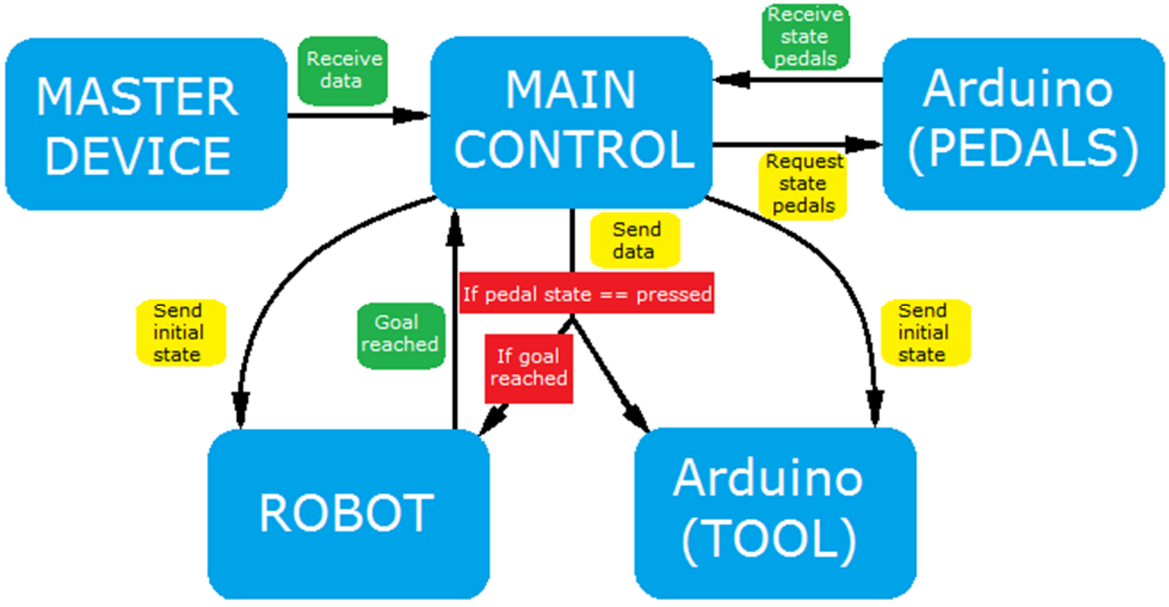


Figure 45. Scheme of communication flow.

When the system starts working, the PC sends the correspond data to the instruments and the

robot control the initial state (initial position). The PC (MAIN CONTROL in the figure) receive data (green signs) from the master device, the, Arduino board and the robot, and send data (yellow signs) to the robot controller and the instrument controller.

The master device is constantly sending the position of each join to the PC, the PC captures it in a thread and asks the Arduino for the pedals state. If the pedal state is pressed, the PC will proceed to send data to the instrument's controller. If the robot's controlled has already sent to the PC that the robot reached the target position, the PC also sends the corresponding data to the robot controller.

7. EXPERIMENTAL DESIGN

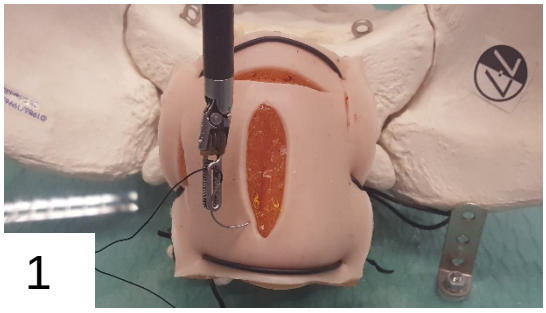
In order to test the functionality of the surgical instrument, an experimental test has been designed. It consists of a suturing task performed in a phantom scenario, which simulates a human pelvis with an open bladder to be sutured (Figure 43). The used tissues in this test simulate the mechanical properties of a real bladder tissues.

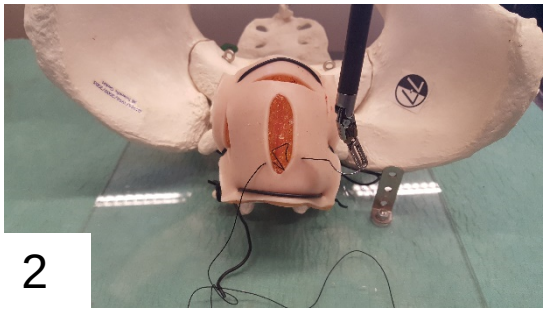
The robot using the new instrument is expected to guide a curved surgical needle during the suture, the trajectory of which requires continuous orientation changes in short distances. The objective of this test is to demonstrate that the surgical system has improved in dexterity, being now able to perform tasks which were impossible to manage with the previous configuration.

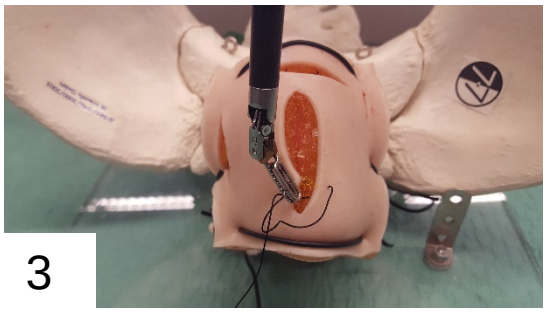
This chapter shows the follow up of a complete suture and mentions, according to the user experience, what points have been critical along the task. The criticisms made by the user give rise to further works proposals. The experimental design has consisted of 3 trials of the same suturing technique (continuous suturing) with satisfactory results in all repetitions. Figure 44 shows the third trial step by step. Each experiment started with the robot oriented and positioned manually to the initial configuration, being the rest of the suture strictly teleoperated through the master device.

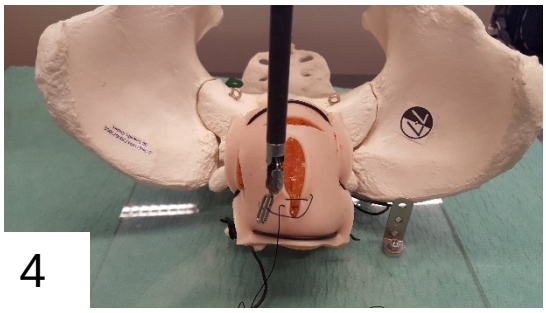
Figure 46. Experimental scene.

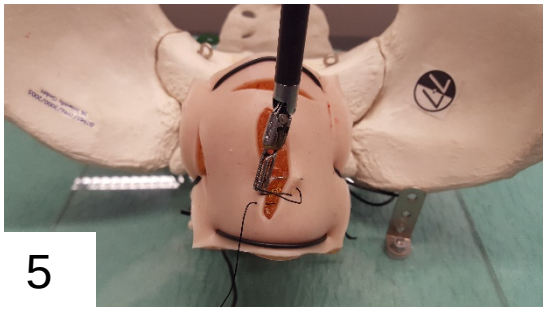
The following sequences of the images shows the different steps to achieve the bladder suturing.

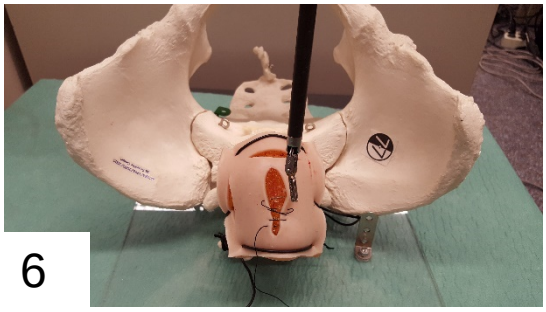

1

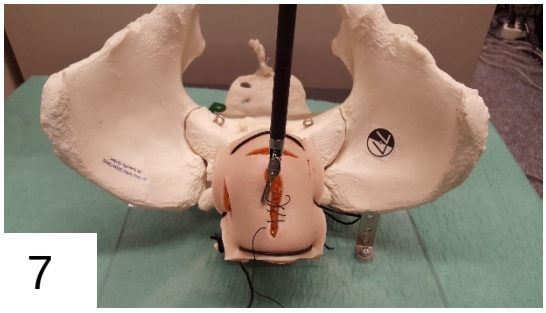

2

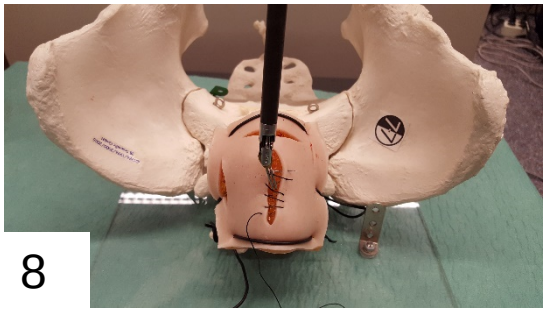

3

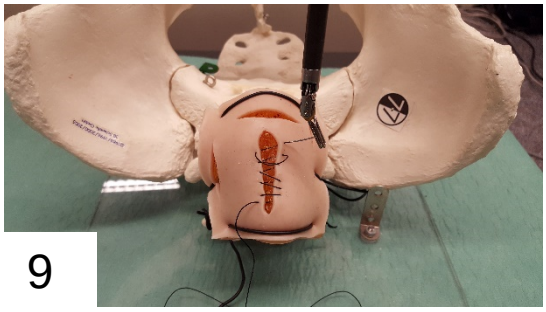

4

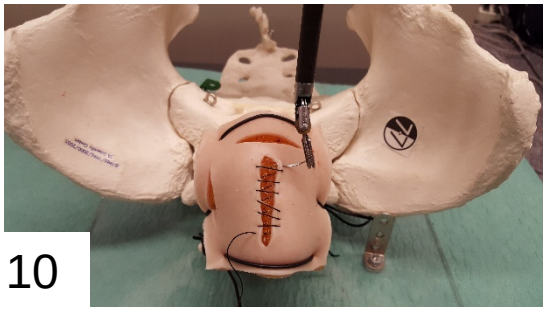

5


6


7


8


9


10

FINAL RESULTS AND CONCLUSIONS

This chapter, the results obtained during the experimental test are summarized and evaluated. In addition to the tests measurements, additional criteria have been taken into account to formulate the conclusions of this project at the end of the chapter.

Achieved Objectives and Results

The most relevant achievement of this project is the improvement of the robots dexterity in performing surgical tasks. In addition to this, the new coupling between the robot and the tool allows a much faster and easier change of instruments.

The coupling is also designed, so that the stem of the tool is aligned with the Z axis the robot end effector (Figure 47). This configuration reduces the complexity of calculating the system's inverse kinematics of the robot and the tool. This alignment has be not fit perfectly in our development, and it is attributable to the manufacturing technique (accuracy and materials provided by a 3D printer) plus the complex geometry of the tool. This problem, can be solved by modifying the materials and design of the instrument support piece, so that the alignment complies with the desired precision.

Figure 47. Alignment tool and link 6 of the robot.

Another expected improvement of the new surgical instrument was to overcome the limited range of orientations provided by a conventional one. In order to compare this improvement the Table 6 shows the difference between the orientation ranges available with the 1 DOF with the 3 DOF surgical instruments. It's worth mentioning that the measurements obtained with the conventional tool required the robot's movements to change its orientations, while in the robotic instrument only the tool tip is moved.

As this Table 6 shows, a significant improvement for orientations alpha and beta has been achieved (increases of 1.5 and 5.1 times the orientation respectively). These values

demonstrate the good capabilities of the implemented tool, which will be translated in less joint displacements of the robot.

Table 6. Comparative between degrees range for each tool orientation. Max conventional orientation vs robotic tool orientation.

	Conventional 1 DOF Tool (max)	3 DOF Tool (robot fix)	INCREASE
alpha	116	180	1.5
beta	35	180	5.1
gamma	540	300	0.6

Although the results are more than acceptable, they could be improved if the orientation of the robotic instrument included the kinematics of the robot, as it is explained in FURTHER WORK. In this case, we can estimate which would be the real comparative between the maximum orientations achieved by both instruments. To estimate these values, it has been considered that the robot reaches the same position limits in both cases, only that the robotic instrument can add its additional DOF to the orientations achieved by the robot end effector itself (Table 7). As expected, the results obtained with the 3 DOF instrument becomes much superior.

Table 7. Comparative degree range for each tool orientation. Max conventional orientation vs Max new tool orientation

	Conventional 1 DOF Tool (max)	3 DOF Tool (max)	% INCREASE
alpha	116	296	2.6
beta	35º	215	6.1
gamma	540	840	1.5

As showed in the EXPERIMENTAL DESIGN chapter, a bladder suturing has been performed. The fact that inverse kinematics of the robot has not been taken into account to orientate the tool, makes the task control become more complicated. Nevertheless, the suture could be carried out and no symptoms of lack of strength were observed when grasping the needle and passing it through the tissue, or when pulling the needle out the stitches.

On the other hand a significant buckling is observed when performing some efforts at the tip of the tool and perpendicular to it. Since the tool shaft is about 460 mm long, it generates a torque which bends the entire tool.

Conclusions

According to the obtained results, the first conclusion to be drawn is that this project has succeeded to improve the existing tool-robot system. A much wider range of available orientation at the tip of the tool has been achieved. Consequently, more complex tasks in the laboratory can be performed, which were formerly impossible.

Besides the wider range of orientations in a reduced space, the control of the tool has appeared not to be optimal. The orientation of the tool tip is achieved solely by the 3 DOF of the instrument, but the 6 DOF of the robot should be also taken into account to replicate the orientations of the master device. The computing of the required inverse kinematics would have definitely helped performing the experimental test more smoothly. The kinematic chain should also include the configuration of the surgical instrument, since it is also composed by links and joints, disregarded in this project.

Additional issues have come up because the reference frames of the master device and the tool tip are not corresponded by any homogeneous transform. The only systems referenced have been the base of the master device with respect to the base of the robot. This solution is valid only for the tools translation, but fail in orientations. An additional transform matrix should be found to close the kinematic chain between the user wrist (master device) and the tool tip.

The selected servos have met our requirements, since it has not been observed any lack of strength during the experimental suture. The needle has never slipped from the instrument's clamps, neither passing through the tissue or being pulled out. Although the actuators were expected to behave properly due to their specifications, the lack of strength was one of the main fears of this project.

A significant buckling was detected when perpendiculars efforts were made to the tip of the tool. This problem could be solved if the coupling parts were manufactured differently or using other materials. The coupling parts used in this project have been printed with a conventional 3D printer, so their mechanical properties are not optimal fixing purposes.

As a summary of this project, it can be determined that the objectives of the project have been achieved. The tool receives the guidelines that are sent from the *Phantom Haptic Device* and the tool plays with a very high speed, so it gives no sense of delay and it gives a very good feeling. Also the movement reproduced by the tip of the tool is very smoothly.

The work done in this project is the first step to start working with new surgical instrument. Nevertheless, it is important to consider the proposed improvements in order to achieve an optimal control of the surgical task.

FURTHER WORK

This project's contribution is the beginning of an encouraging improvement for a research environment which is prepared for a laparoscopic surgical interventions. Down below, some outstanding tasks focused to achieve a better performance of the current system are proposed:

- The implementation of the tool orientation considering the inverse kinematics for the whole robot and surgical instrument (6 + 3 DOF). In order to have the full control of tool the kinematic study should be carried out.
- To manufacture the instrument support in a rigid and resistant material such as metal. A better fixing capacity would avoid movements in the housing, which are translated into vibration at the end of the shaft.
- Scale this project to the two robots and integrate all parts of a teleoperation system. To reproduce a teleoperate surgical interventions, this project should be scaled to control 2 robots with its new robotic surgical instrument. Also a vision system and monitoring should be implemented.

THANKS

First, I would like to thank Alícia Casals, director of this project, the trust placed in me for the preparation of this project. Thanks to her I have had the opportunity to stay in touch and work in robotic applications.

Eduard Bergés, is the person I owe my sincere thanks. He is the one who has been dedicating all the time to help me and guide me throughout this project. I would also like to mention the whole lab team who have contributed their bit during these months when countless doubts and solve problems encountered during the project. Thanks Xavier, Albert and especially Joan Basomba.

As always, I would like to acknowledge the support received from my parents. I especially would like to thank Alina Lucea who has been a staunch personal support and to whom I owe all my love.

REFERENCES

[1] "Wikipedia," 1 March 2016. [Online]. Available: https://en.wikipedia.org/wiki/Laparoscopic_surgery.

[2] J. Dwivedi, I. Mahgoub, "Robotic Surgery - A Review on Recent advances in," in *Florida Conference on Recent Advances in Robotics*, Boca Raton, Florida, 2012.

[3] P. Liverneaux, E. Nectoux, C. Taleb, "The future of robotics in hand surgery," *Elsevier Masson SAS,* vol. Chirurgie de la main, no. 28, pp. 278-285, 2009.

[4] P. Gomes, "Surgical robotics: Reviewing the past, analysing the present,," *ELSEVIER,* Vols. Robotics and Computer-Integrated Manufacturing, no. 27, p. 261–266, 2011.

[5] S. Mosafer, S. Najarian, S. Hajizadeh, N. Simforoosh, "Design motorized hand held flexible instrument for Minimally Invasive Surgery (MIS)," *IEEE Symposium on Industrial Electronics and Applications,* 2009.

[6] C. Preusche, T. Ortmaier, G. Hirzinger, "Teleoperation concepts in minimal invasive surgery," *ELSEVIER,* vol. Control Engineering Practice, no. 10, p. 1245–1250, 2002.

[7] "RC Helicopter Fun," [Online]. Available: http://www.rchelicopterfun.com/rc-servos.html.

[8] "ROBOTIS," [Online]. Available: http://en.robotis.com/index/.

[9] ROBOTIS, "Tech Support ROBOTIS DYNAMIXEL XL-320," 25 3 2016. [Online]. Available: http://support.robotis.com/en/techsupport_eng.htm#product/dynamixel/xl-320.htm.

[10] S. Electrtonics, 5 10 2015. [Online]. Available: http://savageelectrtonics.blogspot.com.

[11] "Black Box," 26 3 2016. [Online]. Available: http://www.blackbox.es/en-es/Page/25469/Technical/Black-Box-Explains/interface-standards/ieee-1394--firewire?gclid=Cj0KEQjw5ti3BRD89aDFnb3SxPcBEiQAssnp0qNIoy96Q5yOXcqclcLA8tPbZ4hfcHQTTK5CS0Eu7fcaAoRz8P8HAQ.

[12] Geomagic, "dentsable," 26 3 2016. [Online]. Available:

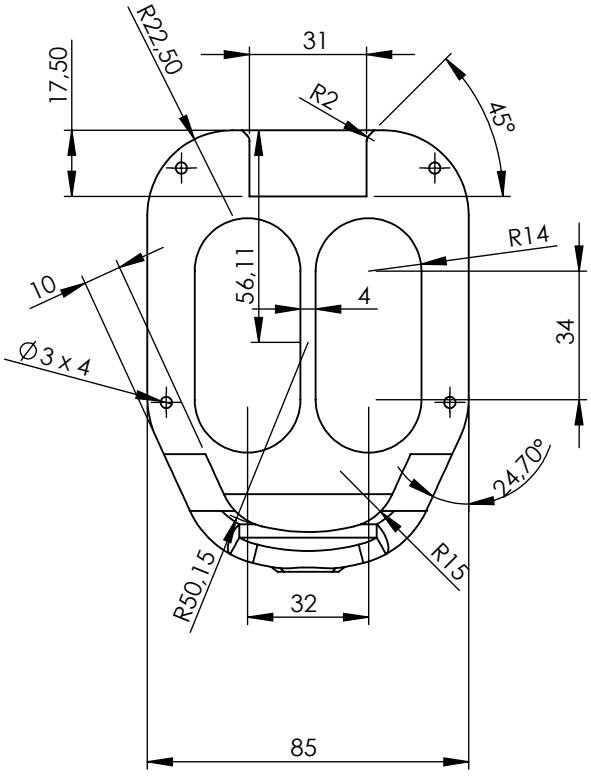
http://www.dentsable.com/haptic-phantom-omni.htm.

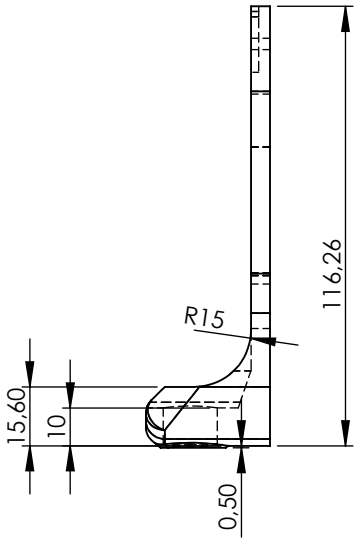
[13] "Geomagic," 28 3 2016. [Online]. Available: http://www.geomagic.com.

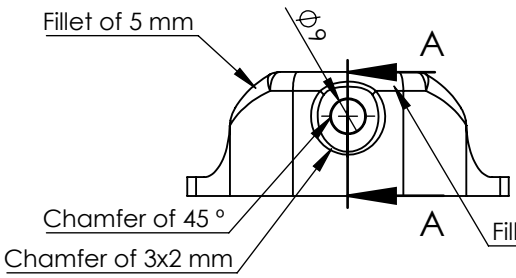
[14] P. D. Team, "SIGVerse," [Online]. Available: http://www.sigverse.org/wiki/en/index.php?Using%20Phantom%20Omni%20Haptik%20Device#ef7886c1.

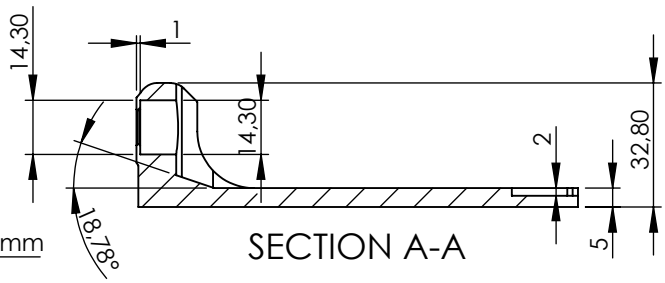
ANNEXES

Mechanical drawings



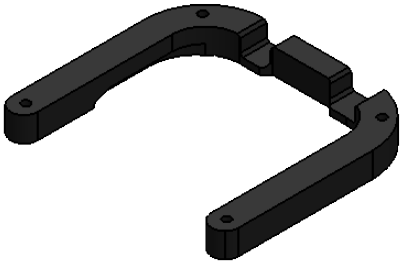


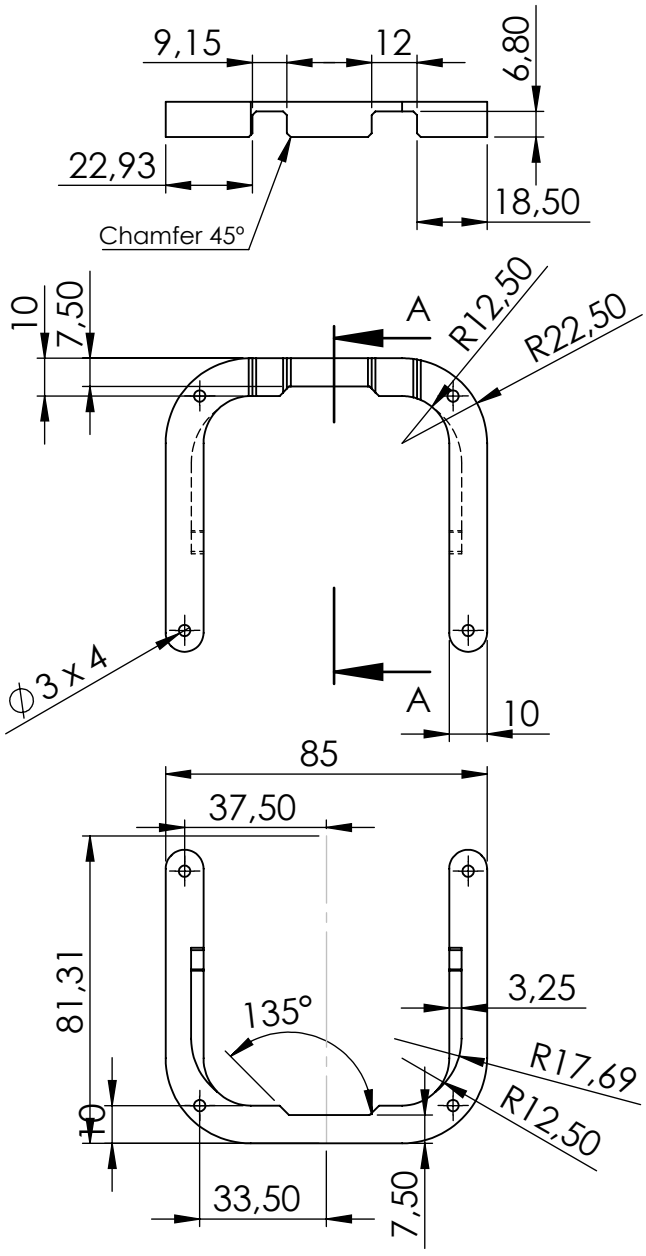


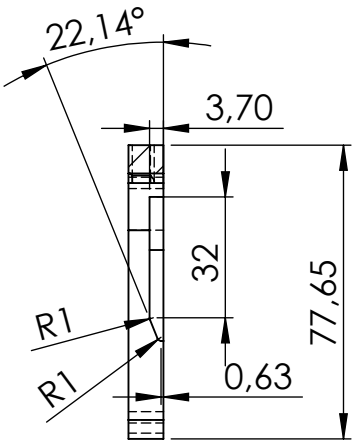


MANUFACTURING 3D PRINTER		**UNLESS OTHERWISE INDICATED:** DIMENSIONS ARE IN MM	**TITLE** Clutch
3D PRINTER CONFIG.: ∅ NOZZLE: 0.3 MM LAYER HEIGHT: 0.25 MM			ETSEIB

	NAME	SIGNATURE	DATE	MATERIAL:	N.º DRAW	
DIBUJ.	SERGIO SÁNCHEZ		27-04-2016	ABS	1	A4
				WEIGHT: 33 g	SCALE:1:2	SHEET 1 OF 1



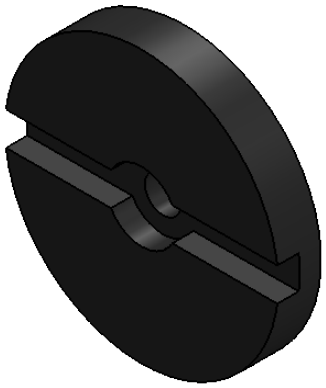


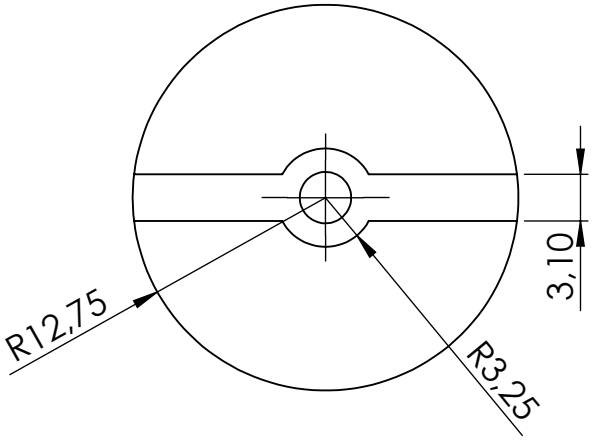


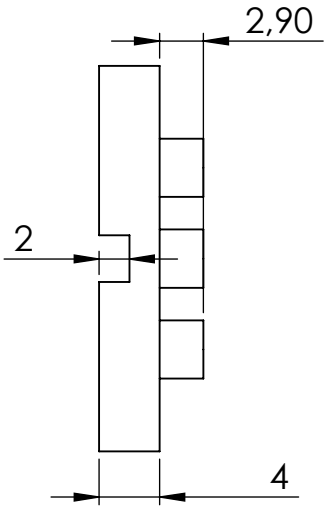
SECTION A-A

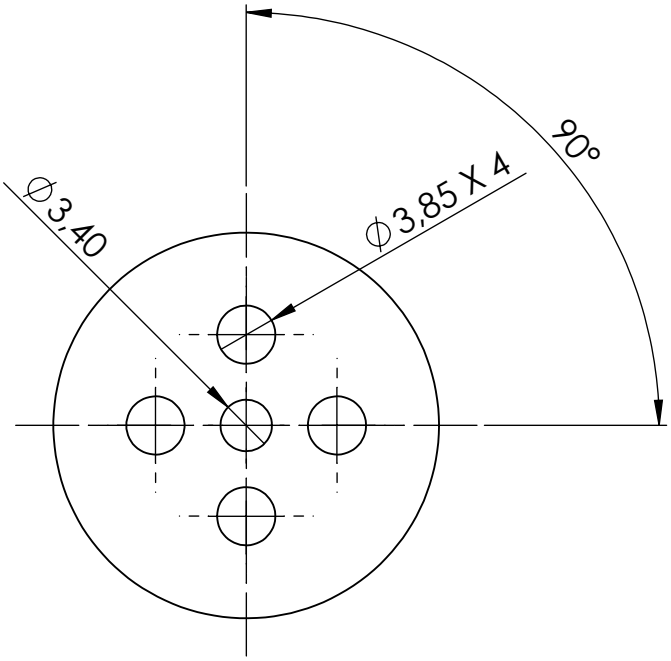
MANUFACTURING 3D PRINTER		**UNLESS OTHERWISE INDICATED:** DIMENSIONS ARE IN MM	**TITLE** Fixing
3D PRINTER CONFIG.: ∅ NOZZLE: 0.3 MM LAYER HEIGHT: 0.25 MM			ETSEIB

	NAME	SIGNATURE	DATE	MATERIAL:	N.º DRAW	
DIBUJ.	SERGIO SÁNCHEZ		27-04-2016	ABS	2	A4
				WEIGHT: 11 g	SCALE:1:2	SHEET 1 OF 1



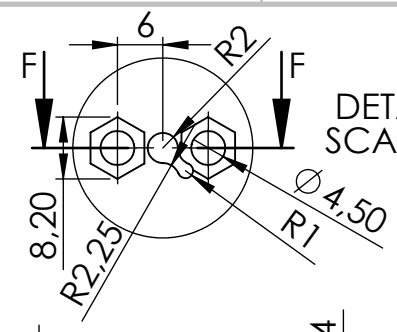




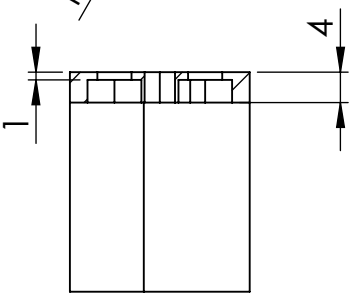


MANUFACTURING 3D PRINTER		**UNLESS OTHERWISE INDICATED:** DIMENSIONS ARE IN MM	**TITLE** Coupling pulley
3D PRINTER CONFIG.: ∅ NOZZLE: 0.3 MM LAYER HEIGHT: 0.25 MM			ETSEIB

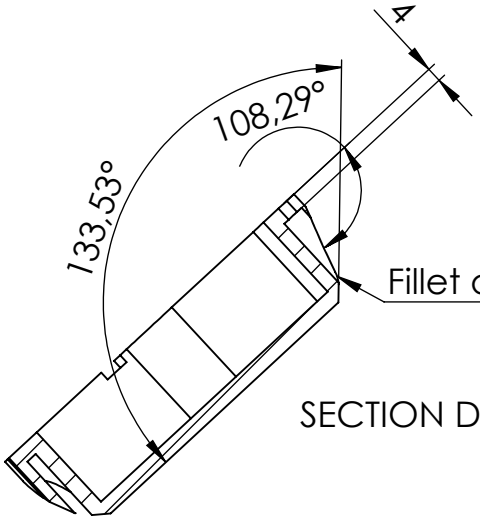
	NAME	SIGNATURE	DATE	MATERIAL:	N.º DRAW	
DIBUJ.	SERGIO SÁNCHEZ		27-04-2016	ABS	3	A4
				WEIGHT: 2 g	SCALE:2:1	SHEET 1 OF 1



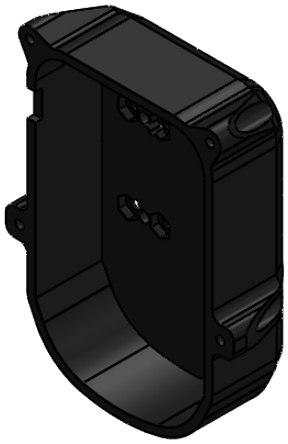
DETAIL A
SCALE 1:1

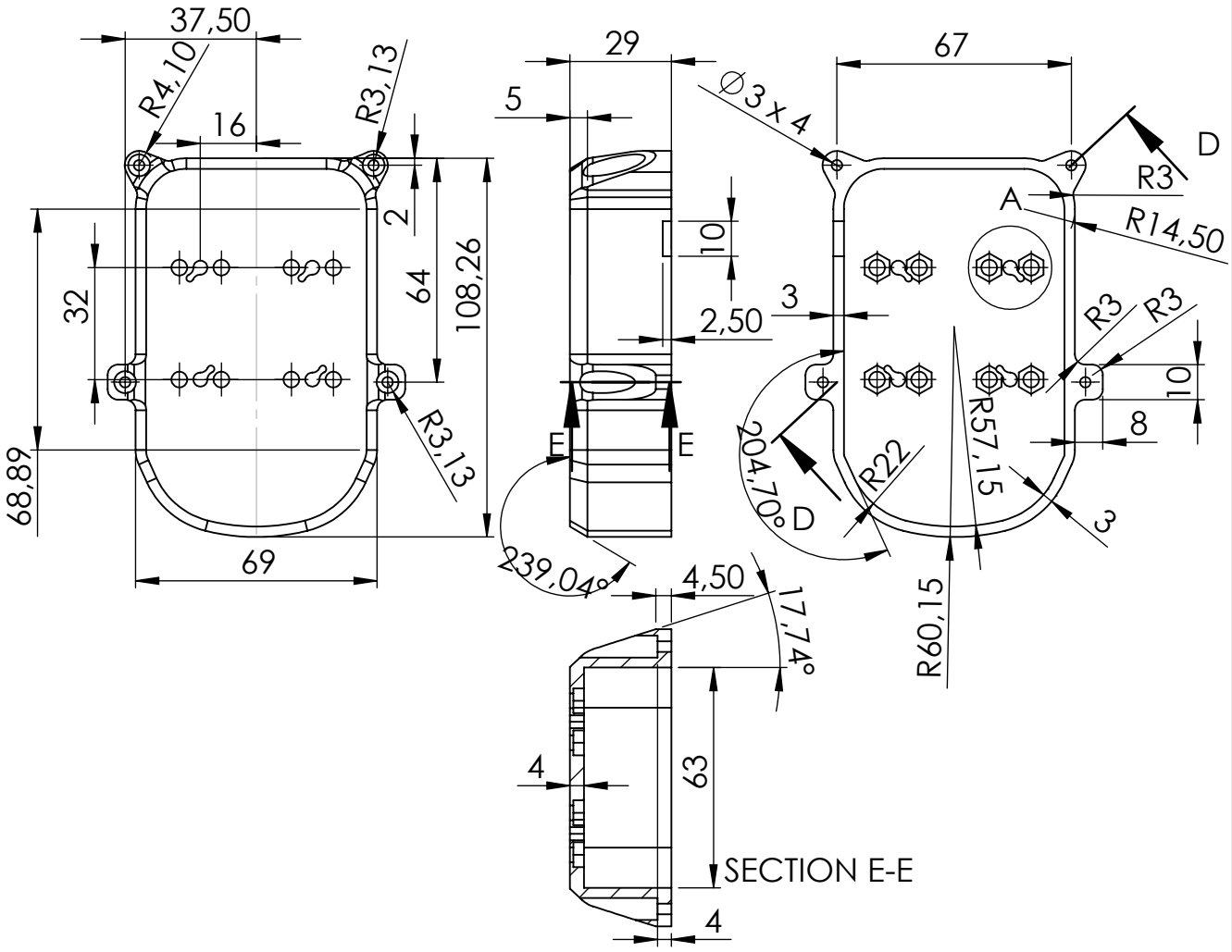


SECTION F-F
SCALE 1:1



SECTION D-D

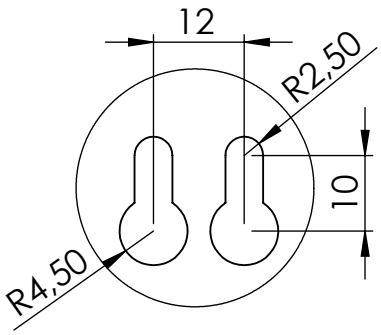




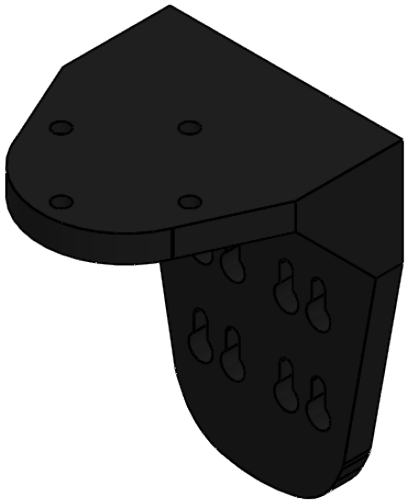
SECTION E-E

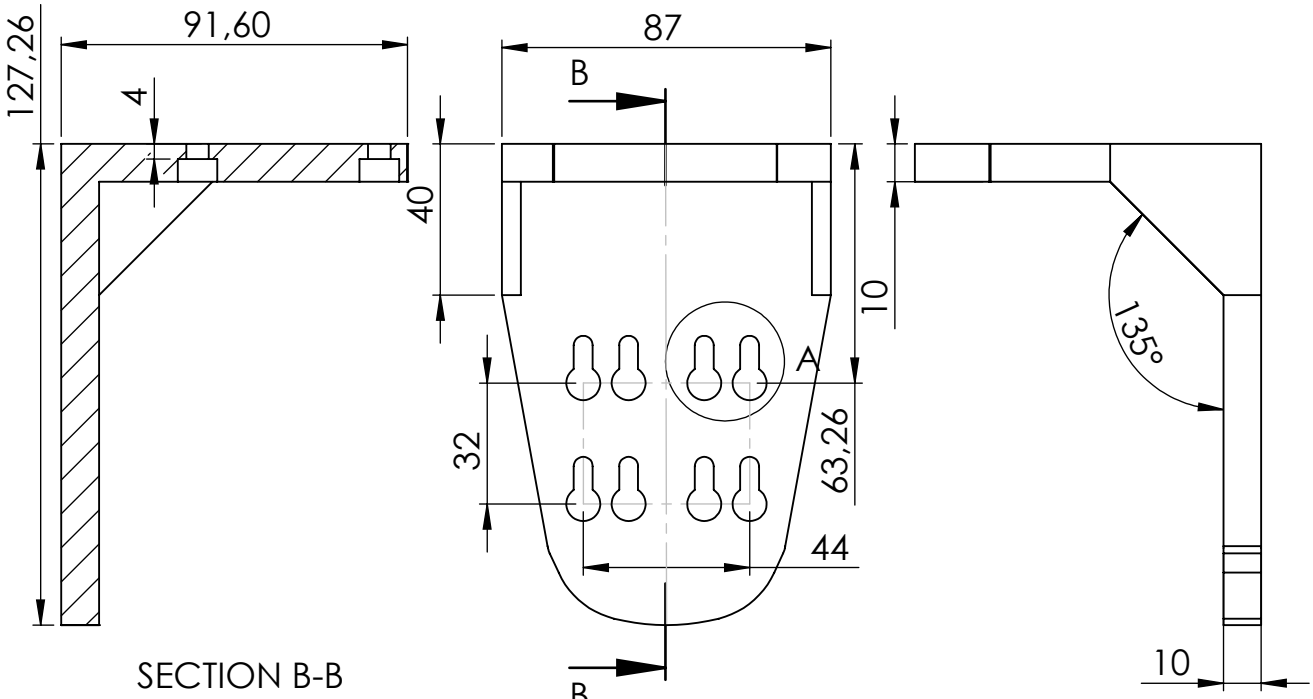
MANUFACTURING 3D PRINTER		**UNLESS OTHERWISE INDICATED:** DIMENSIONS ARE IN MM	**TITLE** Housing Servos
3D PRINTER CONFIG.: Ø NOZZLE: 0.3 MM LAYER HEIGHT: 0.25 MM			ETSEIB

	NAME	SIGNATURE	DATE	MATERIAL:	N.º DRAW	
DIBUJ.	SERGIO SÁNCHEZ		27-04-2016	ABS	4	A4
				WEIGHT: 45 g	SCALE:1:2	SHEET 1 OF 1

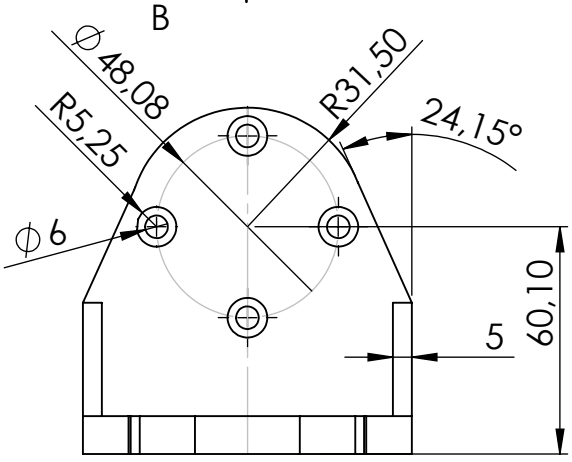


DETAIL A
SCALE 1 : 1





SECTION B-B



MANUFACTURING 3D PRINTER		**UNLESS OTHERWISE INDICATED:** DIMENSIONS ARE IN MM	**TITLE** Tool support
3D PRINTER CONFIG.: Ø NOZZLE: 0.3 MM LAYER HEIGHT: 0.25 MM			ETSEIB

	NAME	SIGNATURE	DATE	MATERIAL:	N.º DRAW	
DIBUJ.	SERGIO SÁNCHEZ		27-04-2016	ABS	5	A4
				WEIGHT: 101 g	SCALE:1:2	SHEET 1 OF 1

Control code

Arduino code

Forceps control and pedal state

```
#include <DynamixelSerial_XL320.h>
boolean debug = false;

int ID = 254; // default ID
int SPEED = 100;
int TORQUE = 100;
int POSITION = 0;


String Id = "";
String Speed = "";
String Torque = "";
String Position = "";

// This pins are used to read the state of the pedals.
int pedalPin_1 = 42;
int pedalPin_2 = 34;

float degree = -1;

String tram = "";             // hold incoming data
boolean completeTram = false;  // whether the tram is complete or not

void setup(){
  // Inicialize the servo at 1Mbps and as a Control Pin 2
  Dynamixel.begin(1000000,2);

  // Initialize serial:
  Serial.begin(115200);

  // reserve 200 bytes for the tram:
  tram.reserve(200);

  //setup pedals
  pinMode(pedalPin_1, INPUT);
  pinMode(pedalPin_2, INPUT);
}

void loop(){

 serialEvent(); // call the function

 if(completeTram){
   processing(tram);
   tram = "";
   completeTram = false;
 }
```

```
}

void serialEvent() {
  while (Serial.available()==0) {}

    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    tram += inChar;

   if (inChar == ';') {
      completeTram = true;
      if(debug){
        Serial.println(tram);
      }
    }

}

void processing(String tram){
  int c = 0;
  unsigned int len = tram.length();
  char buf[len];
  tram.toCharArray(buf, len);

  while(c<len){
    switch(buf[c]){

      case 'I':
        c++;
        Id="";
        while(String(buf[c])!=" " & String(buf[c])!=";" & c<len){
            Id += String(buf[c]);
            c++;
        }
        ID = Id.toInt();
        if(debug){
          Serial.print("ID: ");
          Serial.println(ID, DEC);
        }

      break;

      case 'S':
        c++;
        Speed="";
        while(String(buf[c])!=" " & String(buf[c])!=";" & c<len){
            Speed += buf[c];
            c++;
        }
        SPEED = Speed.toInt();
        Dynamixel.setSpeed(ID, porcentual(SPEED));
        if(debug){
          Serial.print("ID: ");
```

```
      Serial.print(ID, DEC);
      Serial.print("SPEED: ");
      Serial.println(SPEED, DEC);
    }

  break;


  case 'T':
    c++;
    Torque="";
    while(String(buf[c])!=" " & String(buf[c])!=";" & c<len){
          Torque += buf[c];
          c++;
    }
    TORQUE = Torque.toInt();
    Dynamixel.setMaxTorque(ID, porcentual(TORQUE));
    if(debug){
      Serial.print("ID: ");
      Serial.print(ID, DEC);
      Serial.print("TORQUE: ");
      Serial.println(TORQUE, DEC);
    }

  break;


  case 'P':
    c++;
    Position="";
    while(String(buf[c])!=" " & String(buf[c])!=";" & c<len){
        Position += buf[c];
        c++;
    }
    POSITION = Position.toInt();
    Dynamixel.move(ID, fromDegree(POSITION));
    if(debug){
      Serial.print("ID: ");
      Serial.print(ID, DEC);
      Serial.print(", POSITION: ");
      Serial.println(POSITION, DEC);
    }

  break;

  case 'X':

    int pedal_1 = digitalRead(pedalPin_1);
    int pedal_2 = digitalRead(pedalPin_2);

    Serial.print(pedal_1);
    Serial.println(pedal_2);

  break;
```

```
    }

   c++;
  }

}

float porcentual(float p){
  float v=p*1023/100;
  if(v>1023){
    v = 1023;
  }
  if(v<0){
    v = 0;
  }
  return v;
}

float toDegree(float degree){
  float degreeOut=degree*300/1023;
  return degreeOut;
}

float fromDegree(float degree){
  float degreeOut=degree*1023/300;
  if(degreeOut>1023){
    degreeOut = 1023;
  }
  if(degreeOut<0){
    degreeOut = 0;
  }
  return degreeOut;
}
```

XL-320 Arduino library

DynamixelSerial_XL320.cpp

```
1.   FUNCION SYNCWRITE.
2.
3.   ***************************************************************************

4.
5.   */
6.  #if defined(ARDUINO) && ARDUINO >= 100  // Arduino IDE Version
7.  #include "Arduino.h"
8.  #else
9.  #include "WProgram.h"
10. #endif
11.
12. #include "DynamixelSerial_XL320.h"
13.
14. // Macro for the selection of the Serial Port
15.
16. #define sendData(args)  (Serial1.write(args))    // Write Over Serial
17. #define availableData() (Serial1.available())    // Check Serial1 Data Availabl
    e
18. #define readData()      (Serial1.read())         // Read Serial1 Data
19. #define peekData()      (Serial1.peek())         // Peek Serial1 Data
20. #define beginCom(args)  (Serial1.begin(args))    // Begin Serial1 Comunication

21. #define endCom()        (Serial1.end())          // End Serial1 Comunication

22.
23. // Macro for Timing
24.
25. #define delayus(args) (delayMicroseconds(args))  // Delay Microseconds
26.
27. // Macro for Comunication Flow Control
28.
29. #define setDPin(DirPin,Mode)   (pinMode(DirPin,Mode))       // Select the Switc
    h to TX/RX Mode Pin
30. #define switchCom(DirPin,Mode) (digitalWrite(DirPin,Mode))  // Switch to TX/RX
    Mode
31.
32. bool Debug = false;
33. // Private Methods //////////////////////////////////////////////////////////
    //
34.
35. int DynamixelClass::read_error(void)
36. {
37.     Time_Counter = 0;
38.     while((availableData() < 8) & (Time_Counter < TIME_OUT)){  // Wait for Data

39.         Time_Counter++;
40.         delayus(100);
41.     }
42.
43.     while (availableData() > 0){
44.         Incoming_Byte = readData();
45.         if ( (Incoming_Byte == 255) & (peekData() == 255) ){
46.             readData();                                     // Start Bytes
47.             readData();                                     // Start Bytes
48.             readData();                                     // XL-320 ID
```

```
49.             readData();                                         // Length L
50.             readData();                                         // Length H
51.             readData();                                         // Instruction
52.             Error_Byte = readData();                            // Error
53.                 return (Error_Byte);
54.                 Serial.println("Error_Byte");
55.                 Serial.println(Error_Byte, HEX);
56.         }
57.     }
58.     return (-1);
59.     Serial.println("No XL-320 Response");         // No XL-320 Response
60. }
61.
62. // Public Methods //////////////////////////////////////////////////////
63.
64. void DynamixelClass::begin(long baud, unsigned char directionPin)
65. {
66.     Direction_Pin = directionPin;
67.     setDPin(Direction_Pin,OUTPUT);
68.     beginCom(baud);
69.
70.     Serial.begin(115200);    // Begin Serial for debugging
71. }
72.
73. void DynamixelClass::begin(long baud)
74. {
75.     beginCom(baud);
76. }
77.
78. void DynamixelClass::end()
79. {
80.     endCom();
81. }
82.
83. void DynamixelClass::generatePacket(unsigned char ID, int Length, int Mode, int
    Address, int Value)
84. {
85.     char Length_H, Length_L, Address_H, Address_L, Value_H, Value_L;
86.
87.     Length_H = Length >> 8;           // 16 bits - 2 x 8 bits variables
88.     Length_L = Length;
89.
90.     Address_H = Address >> 8;
91.     Address_L = Address;
92.
93.     Value_H = Value >> 8;
94.     Value_L = Value;
95.
96.
97.
98.     unsigned char TxPacket[14] = { XL320_START, XL320_START, XL320_HEADER, XL32
    0_RESERVED, ID, Length_L, Length_H, Mode, Address_L, Address_H, Value_L, Value_
    H};
99.
100.     CRC = update_crc ( 0, TxPacket , Length + 5);
101.     CRC_L = (CRC & 0x00FF);
102.     CRC_H = (CRC>>8) & 0x00FF;
103.
104.     TxPacket[12] = CRC_L;
105.     TxPacket[13] = CRC_H;
106.
107.     switchCom(Direction_Pin,Tx_MODE);
```

```
108.
109.    for(int i=0;i<sizeof(TxPacket);i++) {
110.        sendData(TxPacket[i]);                  // Send Instructions over Serial
111.
112.        if(Debug){
113.            Serial.println("move");
114.            Serial.println(TxPacket[i], HEX);
115.        }
116.    }
117.
118.    delayus(TX_DELAY_TIME);
119.    switchCom(Direction_Pin,Rx_MODE);
120.}
121.
122.int DynamixelClass::move(unsigned char ID, int Position)
123.{
124.    generatePacket(ID, XL320_GOAL_LENGTH, XL320_WRITE_DATA, XL320_GOAL_POSITION
    , Position);
125.    return (read_error());                  // Return the read error
126.}
127.
128.int DynamixelClass::setSpeed(unsigned char ID, int Speed)
129.{
130.    generatePacket(ID, XL320_SPEED_LENGTH, XL320_WRITE_DATA, XL320_GOAL_SPEED,
    Speed);
131.    return (read_error());                  // Return the read error
132.}
133.
134.int DynamixelClass::setMaxTorque(unsigned char ID, int MaxTorque)
135.{
136.    generatePacket(ID, XL320_TORQUE_LENGTH, XL320_WRITE_DATA, XL320_TORQUE_LIMI
    T, MaxTorque);
137.    return (read_error());                  // Return the read error
138.}
139.
140.
141.unsigned short DynamixelClass::update_crc(unsigned short crc_accum, unsigned ch
    ar *data_blk_ptr, unsigned short data_blk_size)
142.{
143.    unsigned short i, j;
144.    unsigned short crc_table[256] = {
145.        0x0000, 0x8005, 0x800F, 0x000A, 0x801B, 0x001E, 0x0014, 0x8011,
146.        0x8033, 0x0036, 0x003C, 0x8039, 0x0028, 0x802D, 0x8027, 0x0022,
147.        0x8063, 0x0066, 0x006C, 0x8069, 0x0078, 0x807D, 0x8077, 0x0072,
148.        0x0050, 0x8055, 0x805F, 0x005A, 0x804B, 0x004E, 0x0044, 0x8041,
149.        0x80C3, 0x00C6, 0x00CC, 0x80C9, 0x00D8, 0x80DD, 0x80D7, 0x00D2,
150.        0x00F0, 0x80F5, 0x80FF, 0x00FA, 0x80EB, 0x00EE, 0x00E4, 0x80E1,
151.        0x00A0, 0x80A5, 0x80AF, 0x00AA, 0x80BB, 0x00BE, 0x00B4, 0x80B1,
152.        0x8093, 0x0096, 0x009C, 0x8099, 0x0088, 0x808D, 0x8087, 0x0082,
153.        0x8183, 0x0186, 0x018C, 0x8189, 0x0198, 0x819D, 0x8197, 0x0192,
154.        0x01B0, 0x81B5, 0x81BF, 0x01BA, 0x81AB, 0x01AE, 0x01A4, 0x81A1,
155.        0x01E0, 0x81E5, 0x81EF, 0x01EA, 0x81FB, 0x01FE, 0x01F4, 0x81F1,
156.        0x81D3, 0x01D6, 0x01DC, 0x81D9, 0x01C8, 0x81CD, 0x81C7, 0x01C2,
157.        0x0140, 0x8145, 0x814F, 0x014A, 0x815B, 0x015E, 0x0154, 0x8151,
158.        0x8173, 0x0176, 0x017C, 0x8179, 0x0168, 0x816D, 0x8167, 0x0162,
159.        0x8123, 0x0126, 0x012C, 0x8129, 0x0138, 0x813D, 0x8137, 0x0132,
160.        0x0110, 0x8115, 0x811F, 0x011A, 0x810B, 0x010E, 0x0104, 0x8101,
161.        0x8303, 0x0306, 0x030C, 0x8309, 0x0318, 0x831D, 0x8317, 0x0312,
162.        0x0330, 0x8335, 0x833F, 0x033A, 0x832B, 0x032E, 0x0324, 0x8321,
163.        0x0360, 0x8365, 0x836F, 0x036A, 0x837B, 0x037E, 0x0374, 0x8371,
164.        0x8353, 0x0356, 0x035C, 0x8359, 0x0348, 0x834D, 0x8347, 0x0342,
```

```
165.         0x03C0, 0x83C5, 0x83CF, 0x03CA, 0x83DB, 0x03DE, 0x03D4, 0x83D1,
166.         0x83F3, 0x03F6, 0x03FC, 0x83F9, 0x03E8, 0x83ED, 0x83E7, 0x03E2,
167.         0x83A3, 0x03A6, 0x03AC, 0x83A9, 0x03B8, 0x83BD, 0x83B7, 0x03B2,
168.         0x0390, 0x8395, 0x839F, 0x039A, 0x838B, 0x038E, 0x0384, 0x8381,
169.         0x0280, 0x8285, 0x828F, 0x028A, 0x829B, 0x029E, 0x0294, 0x8291,
170.         0x82B3, 0x02B6, 0x02BC, 0x82B9, 0x02A8, 0x82AD, 0x82A7, 0x02A2,
171.         0x82E3, 0x02E6, 0x02EC, 0x82E9, 0x02F8, 0x82FD, 0x82F7, 0x02F2,
172.         0x02D0, 0x82D5, 0x82DF, 0x02DA, 0x82CB, 0x02CE, 0x02C4, 0x82C1,
173.         0x8243, 0x0246, 0x024C, 0x8249, 0x0258, 0x825D, 0x8257, 0x0252,
174.         0x0270, 0x8275, 0x827F, 0x027A, 0x826B, 0x026E, 0x0264, 0x8261,
175.         0x0220, 0x8225, 0x822F, 0x022A, 0x823B, 0x023E, 0x0234, 0x8231,
176.         0x8213, 0x0216, 0x021C, 0x8219, 0x0208, 0x820D, 0x8207, 0x0202
177.     };
178.
179.     for(j = 0; j < data_blk_size; j++)
180.     {
181.         i = ((unsigned short)(crc_accum >> 8) ^ data_blk_ptr[j]) & 0xFF;
182.         crc_accum = (crc_accum << 8) ^ crc_table[i];
183.     }
184.
185.     return crc_accum;
186. }
187.
188. DynamixelClass Dynamixel;
```

DynamixelSerial_XL320.h

```
1.   FUNCION SYNCWRITE.
2.
3.   ***************************************************************************

4.  #ifndef DynamixelSerial_XL320_h
5.  #define DynamixelSerial_XL320_h
6.
7.      // EEPROM AREA  ///////////////////////////////////////////////////////

8.  #define XL320_MODEL_NUMBER_L            0  // AX_MODEL_NUMBER_L
9.  #define XL320_MODEL_NUMBER_H            1  // AX_MODEL_NUMBER_H
10. #define XL320_VERSION                   2  // AX_VERSION
11. #define XL320_ID                        3  // AX_ID
12. #define XL320_BAUD_RATE                 4  // AX_BAUD_RATE
13. #define XL320_RETURN_DELAY_TIME         5  // AX_RETURN_DELAY_TIME
14. #define XL320_CW_ANGLE_LIMIT_L          6  // AX_CW_ANGLE_LIMIT_L
15. #define Xl320_CW_ANGLE_LIMIT_H          7  // AX_CW_ANGLE_LIMIT_H
16. #define XL320_CCW_ANGLE_LIMIT           8  // AX_CCW_ANGLE_LIMIT_L
17. #define XL320_CCW_ANGLE_LIMIT_H         9  // AX_CCW_ANGLE_LIMIT_H
18.
19. #define XL320_CONTROL_MODE              11 // NEW OPTION XL320
20. #define XL320_LIMIT_TEMPERATURE         12 // 11  AX_LIMIT_TEMPERATURE
21. #define XL320_DOWN_LIMIT_VOLTAGE        13 // 12  AX_DOWN_LIMIT_VOLTAGE
22. #define XL320_UP_LIMIT_VOLTAGE          14 // 13  AX_UP_LIMIT_VOLTAGE
23. #define XL320_MAX_TORQUE_L              15 // AX_MAX_TORQUE_L  14
24. #define X320_MAX_TORQUE_H               16 // AX_MAX_TORQUE_H  15
25. #define XL320_RETURN_LEVEL              17 // AX_RETURN_LEVEL  16
26.
27. #define XL320_ALARM_SHUTDOWN            18  // AX_ALARM_SHUTDOWN
28.
29.
30.     // RAM AREA  //////////////////////////////////////////////////////////
31. #define XL320_TORQUE_ENABLE             24  // AX_TORQUE_ENABLE
32. #define XL320_LED                       25  // AX_LED
33.
34. #define XL320_D_GAIN                    27 // NEW OPTION
35. #define XL320_I_GAIN                    28 // NEW OPTION
36. #define XL320_P_GAIN                    29 // NEW OPTION
37.
38.
39. #define XL320_GOAL_POSITION             30  // AX_GOAL_POSITION_L
40.
41.
42. #define XL320_GOAL_SPEED                32  // AX_GOAL_SPEED_L
43.
44. #define XL320_TORQUE_LIMIT              35  // AX_TORQUE_LIMIT_H
45.
46. #define XL320_PRESENT_POSITION          37  // AX_PRESENT_POSITION_H
47.
48.
49. #define XL320_PRESENT_SPEED             39  // AX_PRESENT_SPEED_H
50.
51. #define XL320_PRESENT_LOAD              41  // AX_PRESENT_LOAD_H
52. #define XL320_PRESENT_VOLTAGE           45  // AX_PRESENT_VOLTAGE          42
53. #define XL320_PRESENT_TEMPERATURE       46  // AX_PRESENT_TEMPERATURE      43
54. #define XL320_REGISTERED_INSTRUCTION    47  // AX_REGISTERED_INSTRUCTION   44
55.
56. #define XL320_MOVING                    49  // AX_MOVING                   46
57. #define XL320_HARDWARE_ERROR_STATUS     50  // NEW OPTION XL320
```

```
58.
59.
60. #define XL320_PUNCH                     51  // AX_PUNCH_H                   49
61.
62.     // Status Return Levels ////////////////////////////////////////////////
63. #define XL320_RETURN_NONE               0
64. #define XL320_RETURN_READ               1
65. #define XL320_RETURN_ALL                2
66.
67.     // Instruction Set /////////////////////////////////////////////////////

68. #define XL320_PING                      1
69. #define XL320_READ_DATA                 2
70. #define XL320_WRITE_DATA                3
71. #define XL320_REG_WRITE                 4
72. #define XL320_ACTION                    5
73. #define XL320_RESET                     6
74. #define XL320_SYNC_WRITE                131
75.
76.     // Specials ////////////////////////////////////////////////////////////

77. #define OFF                             0
78. #define ON                              1
79. #define LEFT                            0
80. #define RIGTH                           1
81. #define XL320_BYTE_READ                 1
82. #define XL320_BYTE_READ_POS             2
83. #define XL320_RESET_LENGTH              2
84. #define XL320_ACTION_LENGTH             2
85. #define XL320_ID_LENGTH                 4
86. #define XL320_LR_LENGTH                 4
87. #define XL320_SRL_LENGTH                4
88. #define XL320_RDT_LENGTH                4
89. #define XL320_LEDALARM_LENGTH           4
90. #define XL320_SALARM_LENGTH             4
91. #define XL320_TL_LENGTH                 4
92. #define XL320_VL_LENGTH                 6
93. #define XL320_CM_LENGTH                 6
94. #define XL320_CS_LENGTH                 6
95. #define XL320_CCW_CW_LENGTH             8
96. #define XL320_BD_LENGTH                 4
97. #define XL320_TEM_LENGTH                4
98. #define XL320_MOVING_LENGTH             4
99. #define XL320_RWS_LENGTH                4
100.#define XL320_VOLT_LENGTH               4
101.#define XL320_LED_LENGTH                4
102.#define XL320_TORQUE_LENGTH             7 // 4
103.#define XL320_POS_LENGTH                7 // 4
104.#define XL320_GOAL_LENGTH               7 // 5
105.#define XL320_MT_LENGTH                 5
106.#define XL320_PUNCH_LENGTH              5
107.#define XL320_SPEED_LENGTH              7 // 5
108.#define XL320_GOAL_SP_LENGTH            7
109.#define XL320_ACTION_CHECKSUM           250
110.#define BROADCAST_ID                    254
111.#define XL320_START                     255
112.#define XL320_HEADER                    253
113.#define XL320_RESERVED                  0
114.#define XL320_CCW_AL_L                  255
115.#define XL320_CCW_AL_H                  3
116.#define TIME_OUT                        10
117.#define TX_DELAY_TIME                   400
```

```
#define Tx_MODE                         1
#define Rx_MODE                         0
#define LOCK                            1

#include <inttypes.h>

class DynamixelClass {
private:

    unsigned short CRC;

    unsigned char CRC_L;
    unsigned char CRC_H;
    unsigned char CRC_L_R;
    unsigned char CRC_H_R;
    unsigned char Checksum;
    unsigned char Direction_Pin;
    unsigned char Time_Counter;
    unsigned char Incoming_Byte;
    unsigned char Position_High_Byte;
    unsigned char Position_Low_Byte;
    unsigned char Speed_High_Byte;
    unsigned char Speed_Low_Byte;
    unsigned char Load_High_Byte;
    unsigned char Load_Low_Byte;

    int Moving_Byte;
    int RWS_Byte;
    int Speed_Long_Byte;
    int Load_Long_Byte;
    int Position_Long_Byte;
    int Temperature_Byte;
    int Voltage_Byte;
    int Error_Byte;

    int read_error(void);

public:

    void begin(long baud, unsigned char directionPin);
    void begin(long baud);
    void end(void);
    void generatePacket(unsigned char ID, int Length, int Mode, int Address, in
   t Value);


    int reset(unsigned char ID);
    int ping(unsigned char ID);

    int setID(unsigned char ID, unsigned char newID);
    int setBD(unsigned char ID, long baud);

    int move(unsigned char ID, int Position);
    int controlMode(unsigned char ID,unsigned char Status);
    int turn(unsigned char ID, bool SIDE, int Speed);



    int setTempLimit(unsigned char ID, unsigned char Temperature);
    int setAngleLimit(unsigned char ID, int CWLimit, int CCWLimit);

    int setMaxTorque(unsigned char ID, int MaxTorque);
```

```
179.    int setSpeed(unsigned char ID, int Speed);
180.    int LED(unsigned char ID,  char led_color[]);
181.    int setShutdownAlarm(unsigned char ID, unsigned char SALARM);
182.    int P(unsigned char ID,  int P);
183.    int I(unsigned char ID,  int I);
184.    int D(unsigned char ID,  int D);
185.    int setPunch(unsigned char ID, int Punch);
186.
187.    int moving(unsigned char ID);
188.
189.    int readTemperature(unsigned char ID);
190.    int readVoltage(unsigned char ID);
191.    int readPosition(unsigned char ID);
192.    int readSpeed(unsigned char ID);
193.    int readLoad(unsigned char ID);
194.
195.    int torqueStatus(unsigned char ID, bool Status);
196.
197.
198.    unsigned short update_crc(unsigned short crc_accum, unsigned char *data_blk
    _ptr, unsigned short data_blk_size);
199.};
200.
201.extern DynamixelClass Dynamixel;
202.
203.#endif
```

PC code

Phantom Omni Joystick Thread

```
1.  UINT ThreadPhantom( LPVOID pParam )
2.  {
3.      CPhantomDlg* pObject = (CPhantomDlg*)pParam;
4.      HDErrorInfo error;
5.
6.      CString str;
7.      QueryPerformanceCounter( (LARGE_INTEGER*)&initial );
8.
9.      //Init
10.     hdPhantomL = hdInitDevice("Default PHANToM");
11.
12.     if (HD_DEVICE_ERROR(error = hdGetError()))
13.     {
14.         return 0;
15.     }
16.
17.     pObject->ConnPhL = true;
18.
19.
20.     hdSetSchedulerRate(1000);
21.
22.     hdStartScheduler();
23.
24.     if (HD_DEVICE_ERROR(error = hdGetError()))
25.     {
26.         return -4;
27.     }
28.
29.
30.     while( pObject->bThreadClientPhActive )
31.     {
32.         pObject->SetDlgItemInt( IDC_THPHANTOM, msc );
33.         hdScheduleSynchronous(PhantomCallback, &dDataLeft[0], HD_MIN_SCHEDULER_
    PRIORITY);
34.
35.         long long int bits = ((long long int*)dDataLeft)[0];    //Data status i
    nformation from PHANToM
36.
37.
38.         QueryPerformanceCounter( (LARGE_INTEGER*)&final );
39.         QueryPerformanceFrequency( (LARGE_INTEGER*)&freq );
40.
41.         int msc = (final-initial)*1000.0/freq;
42.
43.         //Positions
44.         dPosX_PhL = dDataLeft[2];
45.         dPosY_PhL = dDataLeft[0];
46.         dPosZ_PhL = dDataLeft[1];
47.
48.         pObject->SetDlgItemInt( IDC_POSX_L, (int)dPosX_PhL );
49.         pObject->SetDlgItemInt( IDC_POSY_L, (int)dPosY_PhL );
50.         pObject->SetDlgItemInt( IDC_POSZ_L, (int)dPosZ_PhL );
51.
52.         //Orientacions
53.         dRotA_PhL = dDataLeft[3];
54.         dRotB_PhL = dDataLeft[4];
```

```
55.         dRotG_PhL = dDataLeft[5];
56.
57.         int MA = (int)(dRotA_PhL*180/3.14159);
58.         int MB = (int)(dRotB_PhL*180/3.14159);
59.         int MG = (int)(dRotG_PhL*180/3.14159);
60.
61.         pObject->SetDlgItemInt( IDC_ROTA_L, MA );
62.         pObject->SetDlgItemInt( IDC_ROTB_L, MB );
63.         pObject->SetDlgItemInt( IDC_ROTG_L, MG );
64.
65.         pObject->SetDlgItemInt( IDC_LBUTT1, buttonState );
66.
67.
68.     }
69.
70.     Sleep(300); //Wait until thread is closed
71.     //Set finished thread variable
72.     pObject->bThreadClientPhFinished = true;
73.     return 0;
74. }
```

Arduino Thread

```
UINT ThreadArduino( LPVOID pParam ){
    CPhantomDlg* pObject = (CPhantomDlg*)pParam;

    CSerialPort spSerial;
    bool bSerialIsOpen;

    //Open Lower Master Serial Port
    spSerial.Open( 3, 115200, CSerialPort::NoParity, 8, CSerialPort::OneStopBit
, CSerialPort::NoFlowControl );

    //Verify if serial communication is open
    bSerialIsOpen = spSerial.IsOpen();

    while(spSerial.IsOpen()){
        //char nWSRecData[2];
        char sRxBuf[100];
        int timeout = 0;
        __int64 iFrequency;

        int SizeOfInt = 1;
        int SizeOfSpace = 1;
        int numberOfPedals = 2;
        int TotalBytesRec = SizeOfInt*numberOfPedals+SizeOfSpace*(numberOfPedal
s-1);


        bool bSerialIsOpen;

        //Verify if serial communication is open
        bSerialIsOpen = spSerial.IsOpen();

        //Clear Read and Write buffers
        spSerial.ClearWriteBuffer();
        spSerial.ClearReadBuffer();

        CString szData;
        szData.Format( "X;");
        spSerial.Write(szData.GetBuffer(), szData.GetLength());

        while(spSerial.BytesWaiting()<TotalBytesRec){
        }

        if(spSerial.BytesWaiting() >= TotalBytesRec)
                {
                    DWORD dwReadLA = spSerial.Read( sRxBuf, spSerial.BytesWaiti
ng() );

                        if(sRxBuf[0]=='1')
                        {
                            pObject-> c_pedal1.SetCheck(true);
                        }else
                        {
                            pObject-> c_pedal1.SetCheck(false);
                        }

                        if(sRxBuf[1]=='1')
                        {
                            pObject-> c_pedal2.SetCheck(true);
                        }else
```

```
57.                         {
58.                             pObject-> c_pedal2.SetCheck(false);
59.                         }
60.                     spSerial.ClearWriteBuffer();
61.                     spSerial.ClearReadBuffer();
62.
63.                     PEDAL_1 = pObject-> c_pedal1.GetCheck();
64.                 }
65.
66.             if(PEDAL_1)
67.                 {
68.                     CString szData;
69.                     int X1=150, X2=150, X3=150, X4=150;
70.
71.                     int ZERO_MOTOR = 150;
72.
73.                     int TWEEZERS_OPEN_CLOSE=0;
74.                     switch(buttonState)
75.                     {
76.                         case 0:
77.                             TWEEZERS_OPEN_CLOSE=25;
78.                         break;
79.                         case 1:
80.                         case 2:
81.                             TWEEZERS_OPEN_CLOSE=-20;
82.                             break;
83.                     }
84.
85.                     int MA = (int)(1*dRotA_PhL*180/3.14159);
86.                     int MB = (int)(1*dRotB_PhL*180/3.14159);
87.                     int MG = (int)(1*dRotG_PhL*180/3.14159);
88.
89.                     // Motor transformation
90.                     X1 = -MB + ZERO_MOTOR + TWEEZERS_OPEN_CLOSE;
91.                     X2 = -MB + ZERO_MOTOR - TWEEZERS_OPEN_CLOSE;
92.                     X3 = MG + ZERO_MOTOR;
93.                     X4 = -MA + ZERO_MOTOR;
94.
95.                     szData.Format( "I1 P%d I2 P%d I3 P%d I4 P%d;", X1, X2, X3,
    X4);
96.
97.                     //Clear Read and Write buffers
98.                     spSerial.ClearWriteBuffer();
99.                     spSerial.ClearReadBuffer();
100.
101.                     spSerial.Write(szData.GetBuffer(), szData.GetLength());
102.                 }else{
103.                     dPosX_PhL_old= dPosX_PhL;
104.                     dPosY_PhL_old= dPosY_PhL;
105.                     dPosZ_PhL_old= dPosZ_PhL;
106.
107.                     dRotA_PhL_old= dRotA_PhL;
108.                     dRotB_PhL_old= dRotB_PhL;
109.                     dRotG_PhL_old= dRotG_PhL;
110.                 }
111.     }
112.
113.     //Close
114.     spSerial.Close();
115.     return 0;
116. }
```

Robot Thread

```
1.  void CPhantomDlg::ReceiveIsOK(int id, int nErrorCode)
2.  {
3.
4.      QueryPerformanceCounter( (LARGE_INTEGER*)&final );
5.      QueryPerformanceFrequency( (LARGE_INTEGER*)&freq );
6.
7.      msc = (final-initial)*1000.0/freq;
8.
9.      QueryPerformanceCounter( (LARGE_INTEGER*)&initial );
10.
11.     CString str;
12.
13.     Vec_DP cartesian(6);
14.     Vec_DP joints(6);
15.     int code;       //Operation code send by Robot Controller
16.     int error;      //Error id. send by Robot Controller
17.     CString m_strController;
18.
19.     ////////////////////////////////////////////
20.     //RCS8 receives mssg
21.
22.     if ((id == RCS8) && ConnR0)
23.     {
24.         //Read the incoming message
25.         Clientsock0->Receive(&str);
26.         Interpret::capture_data(str,&cartesian[0],&cartesian[1],&cartesian[2],&
    cartesian[3],&cartesian[4],&cartesian[5],
27.         &joints[0],&joints[1],&joints[2],&joints[3],&joints[4],&joints[5],&erro
    r, &m_strController, &code);
28.
29.             if ((cartesian[0]<0.001)&&(cartesian[1]<0.001)&&(cartesian[2]<0.001
    )&&(cartesian[3]<0.001)&&(cartesian[4]<0.001)&&(cartesian[5]<0.001))
30.             {
31.                 //str =Interpret::incremental_cartesian(0,0,0,0,0,0,MOVE);
32.                 str = Interpret::gestual(0,0,0,0,0,0,MOVE);
33.                 Clientsock0->Send(&str);
34.             }
35.             else
36.             {
37.
38.                 //Display robot values on interface
39.
40.                 dPosX_R1 = cartesian[0]; dPosY_R1 = cartesian[1]; dPosZ_R1 = ca
    rtesian[2];
41.                 dRotA_R1 = cartesian[3]; dRotB_R1 = cartesian[4]; dRotG_R1 = ca
    rtesian[5]; // Cartesianes
42.
43.
44.                 if (error !=0)
45.                 {
46.                     m_CMessageBox.InsertString(0,(LPCTSTR)"Error amb Robot CS8"
    );
47.                 }
48.                 else
49.                 {
50.                     switch(code)
51.                     {
52.                         case MOVE:
53.
```

```
                                if( PEDAL_1 )
                                {
                                    float s = 0.5;
                                    double increment_x = s*(dPosX_PhL - dPosX_PhL_o
    ld);
                                    double increment_y = s*(dPosY_PhL - dPosY_PhL_o
    ld);
                                    double increment_z = s*(dPosZ_PhL - dPosZ_PhL_o
    ld);
                                    int n = 10;
                                    if( increment_x > n ) increment_x = n;
                                    if( increment_x < -n ) increment_x = -n;
                                    if( increment_y > n ) increment_y = n;
                                    if( increment_y < -n ) increment_y = -n;
                                    if( increment_z > n ) increment_z = n;
                                    if( increment_z < -n ) increment_z = -n;

                                    str = Interpret::gestual(-
    increment_y,increment_x,increment_z,0,0,0,MOVE);

                                    dPosX_PhL_old= dPosX_PhL;
                                    dPosY_PhL_old= dPosY_PhL;
                                    dPosZ_PhL_old= dPosZ_PhL;


                                }
                                else
                                {
                                    str = Interpret::gestual(0,0,0,0,0,0,MOVE);
                                }

                                Clientsock0->Send(&str);

                                break;
                            case INIT_POSE:

                              str = Interpret::incremental_cartesian(0,0,0,0,0,0, W
    AIT);
                            Clientsock0->Send(&str);

                            break;

                        case WAIT:
                            TRACE("%s",str);


                            break;

                            default:

                                break;

                        }
                    }
                }
    }
    else
    {
    }
        UpdateData(false);//passa de variable a pantalla
}
```

Project cost

The following cost tables already includes VAT.

Table 8. Project components

	UNITS	COST PER UNIT (€)	COST (€)
Arduino Mega	1	42.98	42.98
DYNAMIXEL XL-320	4	29.90	119.60
SN74LS241N	1	1.89	1.89
PEDAL	1	19.90	19.90
PROTO PCB OF HOLES	1	4.55	4.55
CONNECTORS	4	0.98	3.92
PRINTER PIECE 1	1	5.66	5.66
PRINTER PIECE 2	1	2.22	2.22
PRINTER PIECE 3	4	1.04	4.16
PRINTER PIECE 4	1	5.9	5.9
PRINTER PIECE 5	1	12.02	12.02
SHELL	1	0.25	0.25
BEARING	1	3	3
SCREW M3	4	0.3	1.2
NUT M3	4	0.15	0.6
SCREW M4	8	0.4	3.2
NUT M4	16	0.3	4.8
TOTAL			
			235.85 € [1]

[1] VAT not included

Table 9. Development time.

	HOURS	HOUR COST (€)	TOTAL COST (€)
ENGINEERING	576	55.00	31,680.00[1]

Table 10. Total project cost.

Total project cost	**31,915.85 €** [1]